Departamento de Informática

PROYECTO FIN DE CARRERA

PORTAL-PLATAFORMA DE CULTURA MUSICAL

Autor: ÓSCAR CUEVAS LANCHARES

Tutor: JESÚS HERNANDO CORROCHANO

Leganés, Enero de 2016

Título: PORTAL-PLATAFORMA DE CULTURA MUSICAL
Autor: Óscar Cuevas Lanchares
Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

ÍNDICE

PÁG.

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Agradecimientos

A la **música** por ayudarme en los malos momentos, y en los buenos.

A **Manuel Rodríguez Gonzalo** por darme la idea de que todo esto podía ser un PFC y abrirme el camino para realizarlo.

A mi familia, mi mujer **Nuria** por darme un toque de atención cuando era necesario, mi hija **Aurora** por lo mismo y por existir y mi hijo **Raúl** por darme esperanzas para verle crecer.

A mis **profesores**, tanto del colegio, como del instituto y de la universidad, por todo lo otorgado sin pedir nada a cambio.

A mi tutor **Jesús Hernando Corrochano** por su ayuda y apoyo inestimables.

Resumen

Lo que más caracteriza al ser humano es su cultura y, en los tiempos que corren, es algo "obligatorio" el adaptar los formatos de esa cultura a estos tiempos modernos, utilizando las últimas tecnologías disponibles. Es obvio que desde que apareció la imprenta, la cultura se ha extendido y se ha hecho permanente en el tiempo, pero es necesario adaptar los formatos a la era de Internet, con el acceso inmediato, ya casi desde cualquier lugar; con los teléfonos móviles inteligentes y tabletas; a parte de los ordenadores de sobremesa que son omnipresentes. Es por todo ello por lo que el proyecto permite **compartir** la **cultura** musical **universalmente.**

Este proyecto desarrolla una **plataforma**, donde los usuarios pueden acceder a los **grupos** musicales, sus **discos** y sus **canciones**, así como la inclusión de los mismos (en el caso de los Administradores). Éstos pueden **insertar** los **discos** y **canciones** de los grupos que tengan asignados y relacionarlo con plataformas sociales (en-línea). Además se pueden insertar **frases** célebres (con enlaces) y también **noticias**. Otra posibilidad que se otorga, es registrarse para ser informado de las últimas inclusiones de datos en el sistema.

El sistema es una **aplicación web** que utiliza las técnicas más novedosas y los últimos avances en Internet, como son el uso de HTML5, CSS3, JavaScript, JDO, almacenamiento en la nube, etc.

Han pasado ya **dieciocho años** desde que terminé las asignaturas de la carrera **I.T.I.G.** (Ingeniería Técnica en Informática de Gestión) y todo ha cambiado: llegaron los grados, el plan Bolonia, y he tenido que reciclar y actualizar muchos de mis conocimientos, lo que me ha venido estupendamente (ya que, sobre todo, el hecho es que lo más importante de la universidad es que te enseñan a aprender a aprender por ti mismo).

Palabras clave

cultura, cultura musical, música, letras, canciones, grupos, discos, html, jsp, servlet, bootstrap, css.

Abstract

What characterizes the human being is its culture and in these times, is something "must" adapt the formats that culture to modern times, using the latest technologies available. Obviously, since the printing press, the culture has spread and has become permanent in time, but it is necessary to adapt the formats to the Internet era, with immediate, almost anywhere; with smart phones and tablets; besides desktop computers are ubiquitous. It is for this reason that the project enables universally shared musical culture.

This project develops a platform, where users can access the bands, their albums and songs, as well as the inclusion of the same (in the case of the administrators). These can insert discs and songs from their assigned groups and relate social platforms (on-line). In addition you can insert quotations (with links) and news. Another possibility is granted, it is sign up to be informed of the latest inclusions of data in the system.

The system is a web application that uses the latest techniques and the latest developments on the Internet, such as the use of HTML5, CSS3, JavaScript, JDO, cloud storage, etc.

It's been eighteen years since I finished the subjects of the ITIG (Computer Engineering Management) race and everything has changed: arrived grades, the Bolonia plan, and I had to recycle and upgrade many of my knowledge, what I He has famously come (because, above all, the fact is that the most important of the university is to teach you to learn how to teach yourself).

Estructura de la memoria

Los capítulos de la memoria están estructurados de la siguiente forma:

1 – **Motivación y objetivos**: donde se expone la motivación a la hora de realizar el proyecto, los objetivos que se esperan con él y los medios empleados.

2 – **Estado del arte**: en él, se analizan las metodologías de desarrollo más extendidas y los sistemas similares al proyecto que se pretende desarrollar. Se estructura en tecnología, metodología y negocio.

3 – **Estrategia:** se estructura en subcapítulos:

- **Arquitectura**: se estudian diversas arquitecturas de desarrollo para el sistema y se elige la más adecuada.
- **Diseño**: el diseño de la base de datos y el prototipo de la aplicación, así como el API de llamadas desde el cliente al servidor.
- **Historias de usuarios**: necesidades que el cliente requiere para la aplicación.
- **APIS y bibliotecas externas**: objetos externos que se han usado para dar funcionalidad extra al proyecto.
- **Metodología**: metodología de desarrollo que se ha seguido para implementar el proyecto.
- **Pruebas**: pruebas de aceptación que se han realizado para comprobar el correcto funcionamiento del sistema.

4 – **Gestión del proyecto**: donde se presenta la planificación y el presupuesto para la elaboración de todo el proyecto.

5 – **Conclusiones y líneas futuras**: donde se exponen las conclusiones extraídas tras la realización del trabajo, así como las líneas futuras que seguirá el proyecto.

Glosario de términos

Admin: Administrador. En el presente PFC encargado de administrar la información que contiene la aplicación.

Apache: Servidor web HTTP de código abierto para plataformas Unix y más.

API (Application Programming Interface): Es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

API REST: Representational State Transfer, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. Permite crear servicios y aplicaciones.

Backend: Denominado como el lado del servidor.

BLOB: Binary Large Objects u objetos binarios grandes. Son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica.

Framework: Marco de aplicación o conjunto de bibliotecas orientadas a la reutilización de componentes software para el desarrollo de aplicaciones. Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Gmail: Servicio de correo electrónico de Google.

HTML: HyperText Markup Language o lenguaje de marcas de hipertexto. Hace referencia al lenguaje de programación para realizar páginas webs.

HTTP: HiperText Transfer Protocol, orientado a transacciones de la World Wide Web que sigue el esquema petición-respuesta.

ITIG: Ingeniería Técnica en Informática de Gestión.

Java: Lenguaje de programación orientado a objetos.

JavaScript: Lenguaje de programación orientado a objetos.

JDK: Es un software que provee herramientas de desarrollo para la creación de programas en lenguaje Java.

JDO: Capa de persistencia de Java para salvar los datos posteriormente.

JRE: Java Runtime Environment, es un conjunto de utilidades que permiten la ejecución de programas en lenguaje Java.

Metodología: Hace referencia al camino o al conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige un proyecto.

MSIL: Microsoft Intermediate Lenguage, o ahora denominado Common Intermedate Languaje, es el lenguaje ensamblador orientado a objetos en la arquitectura .NET.

MVC, Modelo-Vista-Controlador: Es el patrón de arquitectura software que separa la lógica del negocio y los datos de la interfaz de presentación.

MySQL: Sistema de gestión de base de datos racional, multihilo y multiusuario, ofrecida bajo la GNU GPL.

Open Source: Expresión conocida para software o hardware distribuido y desarrollado libremente.

Patrones de Diseño: Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

PFC: Proyecto Fin de Carrera

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor): Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Plugin: Denominado como complemento, es una aplicación que se agrega a otra para una mayor funcionalidad.

Responsive: (Diseño Adaptativo) es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla.

Scrum: Metodología de desarrollo ágil más extendida en el mercado de desarrollo ágil.

SDK: Software Development Kit o kit de desarrollo de software.

Servidor: Aplicación que recibe peticiones de un cliente y tramita respuestas hacia él.

Servlet: Clase Java utilizada para ampliar la funcionalidad del servidor, comúnmente para el tratamiento de peticiones de usuarios.

SMTP: Simple Mail Transfer Protocol, es el protocolo de red usado para el intercambio de mensajes de correo electrónico.

Sprint: En metodologías ágiles, intervalo de tiempo para realizar una entrega a un cliente.

TAB[i], Tablatura, Tablature: Formas de escritura musical especiales para ciertos instrumentos y que, a diferencia de la notación musical corriente, presentan únicamente las posiciones y colocaciones en el instrumento para la interpretación de una pieza, y no las alturas ni las duraciones de los tonos. Debido a que no es necesario tener un conocimiento musical especial, las tablaturas son relativamente fáciles de leer y de entender.

Wikipedia: Autodefinida como enciclopedia libre, políglota, y editada colaborativamente. Administrada por Fundación Wikimedia, organización sin ánimo de lucro financiada con donaciones.

Windows IIS: Internet Information Services, servidor web y conjunto de servicios para Windows como SMTP, HTTP, HTTPS, etc.

XML: Extensible Markup Language, es un lenguaje de carcas desarrollado por World Wide Web Consortium (W3C), utilizado para almacenar datos.

1. CAPÍTULO 1: MOTIVACIÓN Y OBJETIVOS

1.1 MOTIVACIÓN

El PFC me va a permitir probar y llevar la práctica todos los conocimientos adquiridos a lo largo de la carrera, como son bases de datos (persistencia), lenguajes de programación del lado del servidor y del cliente, interactividad con el usuario e interfaz con el mismo, así como redes, hilos, concurrencia, diseño e implementación de estructuras de datos, y tantas otras.

1.2 OBJETIVOS

El **principal objetivo** es crear una **aplicación web** en la que se pueda visualizar información de los **grupos** musicales junto con sus **discos** y sus **canciones**, todos los datos son **insertados en línea** por los administradores, tanto los grupos, como los discos y las canciones. Además, también se pueden insertar **frases célebres** y **noticias**. Y realizar varios tipos de **búsquedas**.

Se pretende crear una forma de **compartir** la **cultura musical**, aunar en un solo lugar todo lo relativo al conocimiento humano (expresado musicalmente) y poder llevarlo a todo el mundo, incluidos los jóvenes que, por un motivo u otro, no se acercan a esa cultura. La música conecta bastante bien con casi todo el mundo, desde niños, jóvenes, mayores y ancianos. La cultura humana y el conocimiento que está en los libros se debe modernizar para llegar a más gente y mejor, de una forma más amena, intuitiva y sencilla. Como posible ampliación se podría llevar a todo tipo de cultura, no sólo la musical, cuadros, libros…

Un punto importante es que el sistema se basa en que la **información** es **aportada** por los **administradores** de cada grupo/s, por lo que se puede incluir gran cantidad de tipos de música, tanto rock, pop, clásica, infantil, etc. También servirá como centro vertebrador para todas las webs o partes de webs que se considere que aporten algo a la cultura musical. Además de incluir todo tipo de música, se buscará intentar recuperar esas canciones populares que ya casi se han perdido (por antiguas), que no tienen casi cabida en las webs típicas. Se intentará aportar a las canciones todas las etiquetas posibles para hacer mejores **búsquedas** y que la **información** sea lo más **completa** posible, por ejemplo temas de los que trata la letra, sentimientos que transmite, edad apropiada, etc.

Algo que he intentando aportar con este PFC es que la **cultura** que creo que está en la **música** (y especialmente en el Rock) se **sintetice**. Hay muchas canciones que vienen de poemas, o que son poemas en sí mismas; otras son cuentos que transmiten algún mensaje; muchas otras experiencias de vida que, vividas y sentidas por alguien, al contarlas pueden transmitir verdadera emoción y pueden ayudar a muchas personas, ¿quién no se ha refugiado en una canción alguna vez en su vida?, ya sea para darse cuenta que no era el único que sentía eso que sentía, o para expresar eso que se siente y uno no es capaz de decir con palabras. Por eso hay algunas canciones que son instrumentales, porque no hay palabras que digan lo que transmite la música.

El punto de diferenciación que he buscado entre esta plataforma y las ya existentes, son los **resúmenes** de las **canciones** y las **etiquetas** que tienen: están escritas por administradores que son amantes del grupo en cuestión. ¿Quién va a saber más y se va preocupar más por los detalles, que unos de los mayores fans del grupo?

Además, con las posibilidades de las **redes sociales**, se puede mantener un diálogo y ver distintos puntos de vista de cada canción: lo que para uno significa una cosa, para otro puede ser otra, e incluso para el que escribió la canción otra completamente diferente.

SUB-OBJETIVOS:

- Dar las **gracias** (devolver de alguna forma) a tantos y tantos artistas que han entregado su vida al mundo de la música, desde los grandes de la música clásica, hasta los antiguamente marginados genios del rock.

- También eso de tener un hijo, plantar un árbol y escribir un libro. Me faltaba lo del **libro** y este PFC es algo parecido (la plataforma web, no la memoria).

- Ponerme a prueba para ver si era capaz de **crear** algo **nuevo** usando todo lo aprendido.

1.3 QUÉ SE ESPERA CON ESTE PROYECTO

Espero que este PFC me **recicle** y me ponga al día sobre las **nuevas tecnologías** y **metodologías** más actuales.

Yo terminé la carrera I.T.I.G. en el año **1997** y desde entonces todas estas tecnologías de Internet han avanzado bastante: Lenguajes de programación como Java, HTML 5, CSS, JavaScript avanzado, etc.; Metodologías ágiles, servidores con nuevas y maravillosas posibilidades. Por todo esto he tenido que reciclarme bastante, ya que yo me quedé en que buscar en Internet era como bucear en una gran biblioteca y, en cuanto a cuestiones técnicas, los compañeros de clase y los profesores eran la única posibilidad. Mis conocimientos se limitaban a Html sencillo, Pascal y C; para nosotros no existía ni la OO, ni casi la multitarea, ni tantas y tantas cosas que hoy en día son habituales.

1.4 MEDIOS EMPLEADOS

El presente PFC ha sido realizado utilizando los siguientes medios, tanto físicos, como lógicos.

- **Medios físicos:**

  Ordenadores Portátiles:

  Portátil Acer Aspire V 17 Black Edition, con las siguientes características:
  Intel Core i7-4710HQ 2.50Ghz
  16 GB de RAM
  Almacenamiento SSD 250GB, Disco Duro 2TB.
  Resolución Grafica:1920x1080

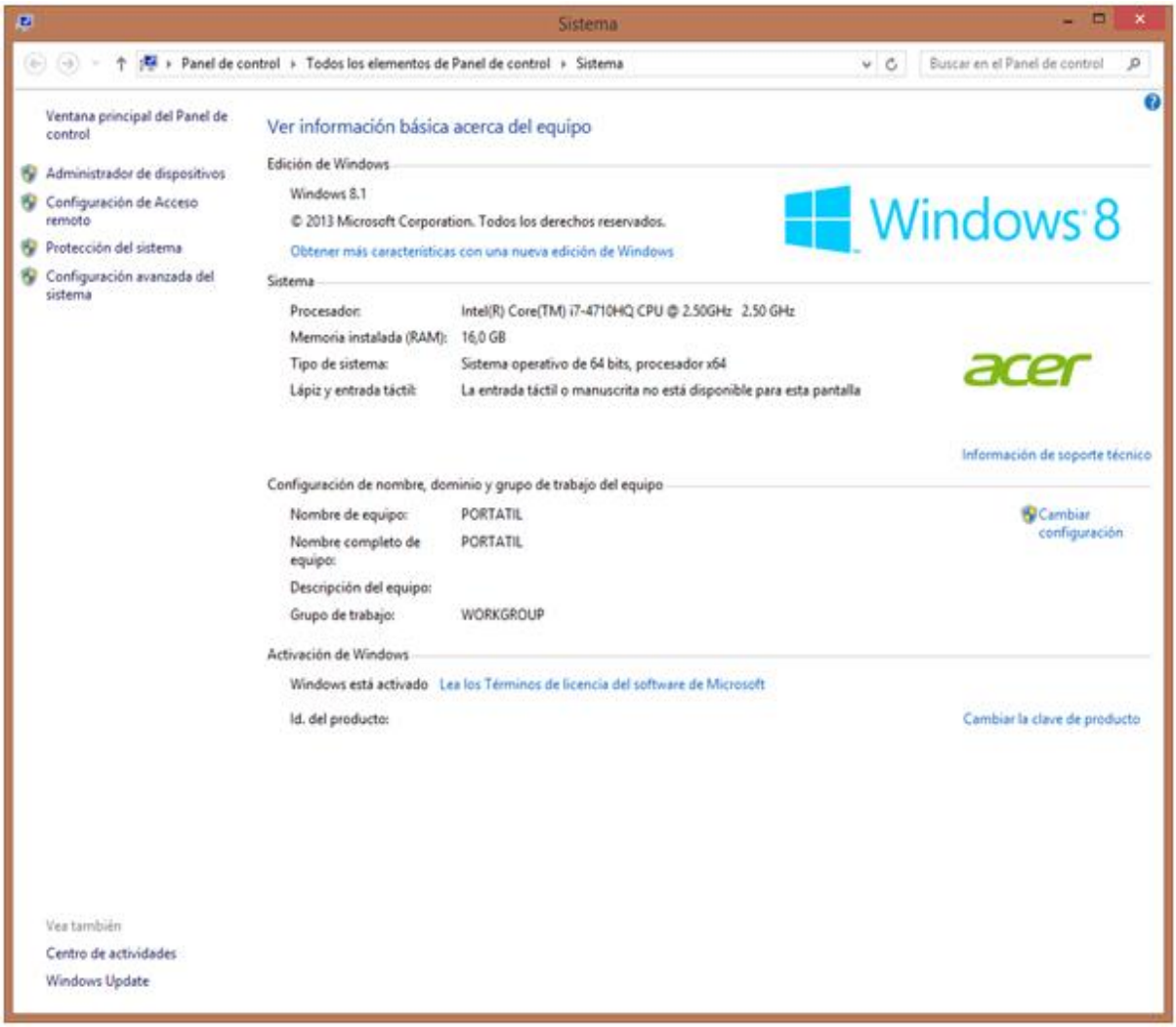


Ilustración 1-1 Características portátil principal para el PFC

Portátil Acer pequeño (miniportátil)

Ordenadores de sobremesa:

Pentium 4 2.80Ghz,1.5 GB RAM 1280x1024
y más, de los que no dispongo de características técnicas.

Móviles:

Samsumg Galaxy Grand 2 800x400 **procesador de doble núcleo a 1.2 GHz** (1GB de RAM)
LG
Y otros tantos en los que he probado, pero no recuerdo sus características.

- **Software:**

  Eclipse Luna
  SDK google app engine para Eclipse Luna.
  Paint
  Navegadores:
    - Firefox
    - Chrome
    - Opera
    - Internet Explorer

  Selenium. Para las pruebas.
  Editor gráfico: https://pixlr.com/editor/

- **Libros:** reflejados en la bibliografía.

- **Internet:** igualmente reflejados en la bibliografía.

- **Lugares:** Universidad Carlos III de Madrid (Aulas informáticas), además de alguna biblioteca pública de Madrid, incluso el bar del pueblo.

2. CAPÍTULO 2: ESTADO DEL ARTE

2.1 TECNOLOGÍA

Como lo que se pretende con esta plataforma es llegar al mayor número de seres humanos posibles, se elegirá una tecnología ampliamente difundida, la tecnología web, aunque se realizará un análisis entre aplicaciones nativas y aplicaciones web. Hoy en día casi todo el mundo (en el primer mundo) accede a internet, desde ordenadores, tablets (tabletas), smartphones (teléfonos inteligentes) y televisiones inteligentes. De ahí que sea una tecnología prioritaria en casi todas las decisiones al realizar aplicaciones de un tipo u otro.

Tabla 2-1 Lista de países por número de usuarios de Internet[ii]

Puesto	País	Usuarios	Penetración (%)	Fecha
—	Mundo	3 035 749 340	42,3 %	2014[1]
001	China	642 261 240	47,4 %	2014
—	Unión Europea	391 395 602	76,5 %	2013
002	Estados Unidos	310 322 257	87,7 %	2014
003	India	243 000 000	19,7 %	2014
004	Brasil	109 773 650	54,2 %	2014
005	Japón	109 626 672	86,2 %	2014
006	Rusia	87 476 747	61,4 %	2014

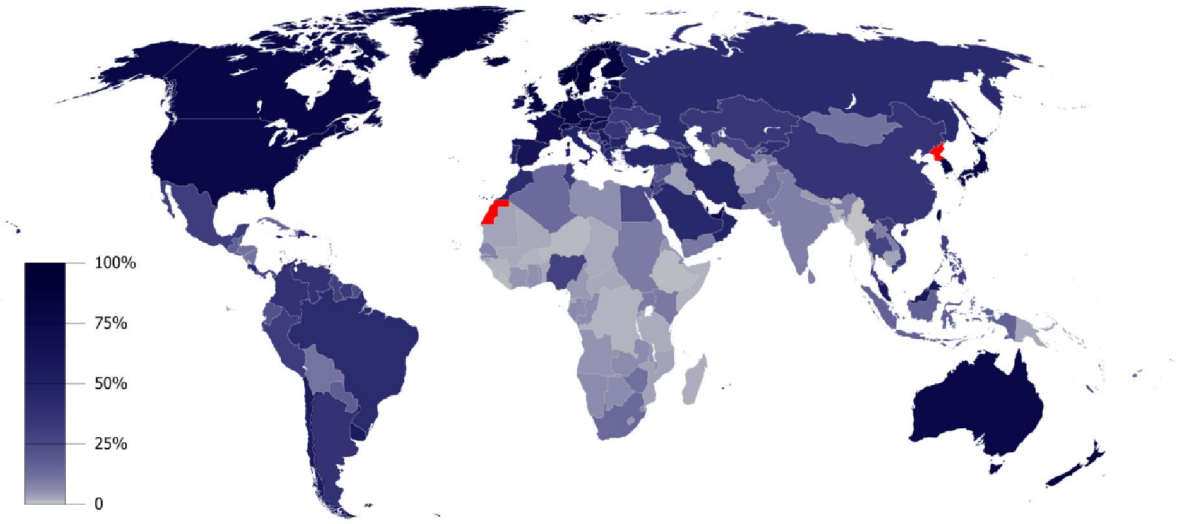


Ilustración 2-1 Penetración de Internet en el mundo[iii]

Ahora se presentan las tecnologías disponibles para el desarrollo del presente PFC.

2.1.1 J2EE

Java Platform, Enterprise Edition o Java EE es un conjunto de estándares y especificaciones para el desarrollo de aplicaciones de empresa basado en lenguaje Java. Este lenguaje está orientado a objetos, es portable, independiente de la plataforma, robusto, multiplataforma y de fácil aprendizaje.

J2EE proporciona todas estas cuestiones técnicas, así como el uso de interfaces de programación para la aplicación (API´s), para el desarrollo y la coordinación de los componentes distribuidos. El estándar J2EE permite el desarrollo de aplicaciones de empresa de una manera sencilla y eficiente. Una aplicación desarrollada permite ser desplegada en cualquier servidor de aplicaciones, con implementación de la especificación J2EE o servidor web que cumpla con el estándar. La arquitectura es la siguiente:

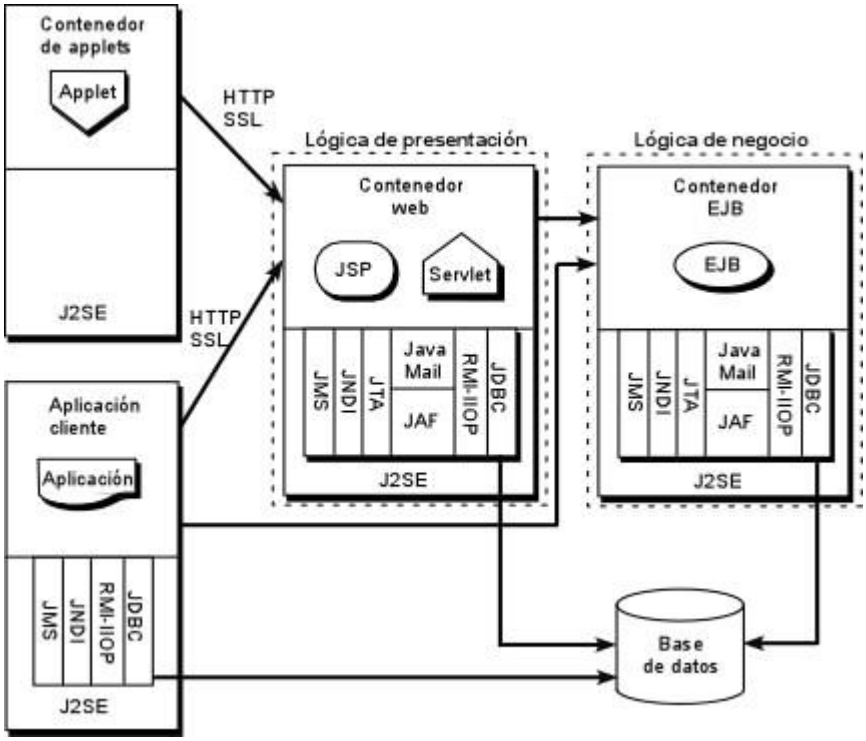


Ilustración 2-2 Arquitectura J2EE

La arquitectura en la aplicación en J2EE consta de 4 capas:

1. Capa cliente: correspondiente con la interfaz gráfica del sistema. Se encarga de interactuar con el cliente.
2. Capa web: localizada en el servidor web, contiene la lógica de presentación para generar la respuesta al cliente. La creación de la respuesta se realiza en el servidor con los componentes Java Servlets, clase utilizada para ampliar funcionalidad del servidor, y JavaServer Pages (JSP), para mostrar dinámicamente un resultado.
3. Capa de negocio: localizada en el servidor, contiene la lógica de negocio. Interactúa con la capa de datos, siendo implementadas como componentes EJB.
4. Capa de datos: responsable de la información de la empresa y su sistema, que incluye la base de datos y su procesamiento.

Con esto, se proporciona una gestión en seguridad, control de transacciones y de concurrencia, gestión en componentes desplegados y asignación de recursos, entre otros.

Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares, ejecutándose sobre un servidor de aplicaciones. Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE.

Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc. y define cómo coordinarlos. Java EE también configura algunas especificaciones

únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), Java Server Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

La última especificación servlet[iv] (la 4.0) incorpora una nueva posibilidad, la web asíncrona, aunque ya apareció en la 3.0[v] . La Web Asíncrona es fundamentalmente diferente, y es esta diferencia la que revoluciona el comportamiento de las aplicaciones web. En la Web Asíncrona es posible entregar al usuario cambios espontáneos en la presentación, a medida que cambia el estado de un sistema dinámico, sin la necesidad de que el usuario interactúe con la interfaz.

Las ventajas son obvias, ya que ahora podemos mantener una representación exacta del sistema en el navegador del usuario. Hay varios ejemplos, como cualquier sistema que brinda una vista de un sistema dinámica, como un portfolio de acciones, un inventario, o un calendario. Cuando hay muchos usuarios interactuando en el mismo sistema, las interacciones de un usuario pueden impactar espontáneamente lo que ven otros usuarios, creando así un sistema verdaderamente colaborativo (la esencia de la Web 2.0).

Nuevamente, hay varios ejemplos, como un cliente de chat simple, o una subasta por eBay. En última instancia, la mayoría de los sistemas con los que interactúan los humanos son colaborativos por naturaleza, por lo que también debería serlo la interfaz que utilizan.

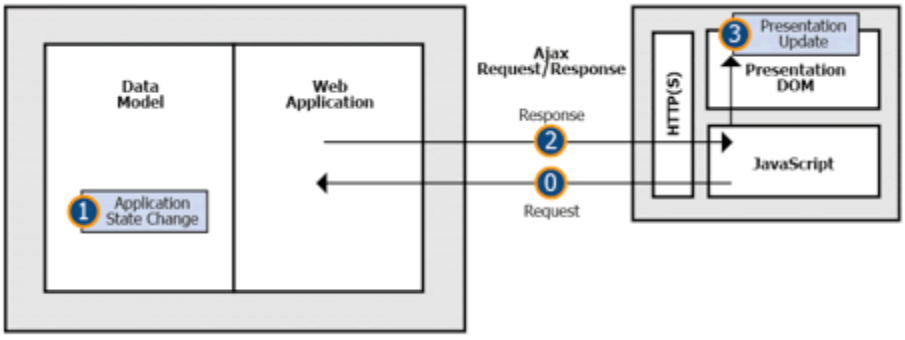


Ilustración 2-3 Web Asíncrona

2.1.2 PHP

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web, que puede ser incrustado en HTML desarrollado bajo una arquitectura cliente-servidor. Es considerado como modelo de aplicación distribuida, donde las tareas se reparten entre cliente, que realiza peticiones, y el servidor, que las responde. Se muestra la arquitectura en la siguiente imagen:

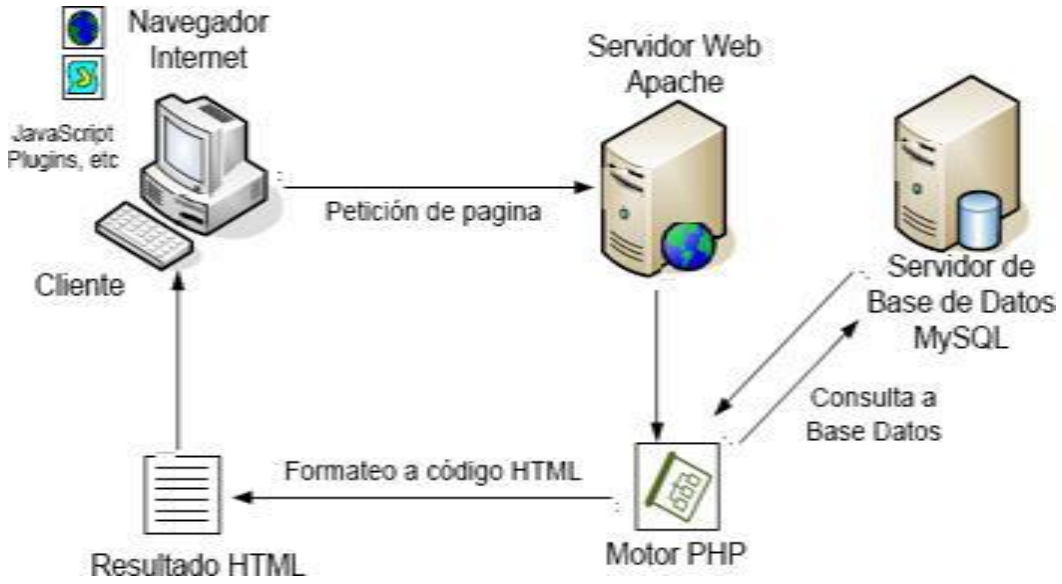


Ilustración 2-4 Arquitectura PHP

Está orientado al desarrollo de aplicaciones web dinámicas que hagan uso de la información de la base de datos. Sencillo y veloz, permite crear programas que se ejecuten en cualquier servidor desde un navegador web, respondiendo a las peticiones ocasionadas por el usuario. Su código fuente es invisible al navegador y al cliente, ya que envía el resultado HTML desde el servidor, incrementando la seguridad. Tiene manejo de excepciones desde su versión PHP5, con capacidad de expandirse con el uso de módulos, como los marcos de trabajo de código abierto, los cuales se muestran más adelante.

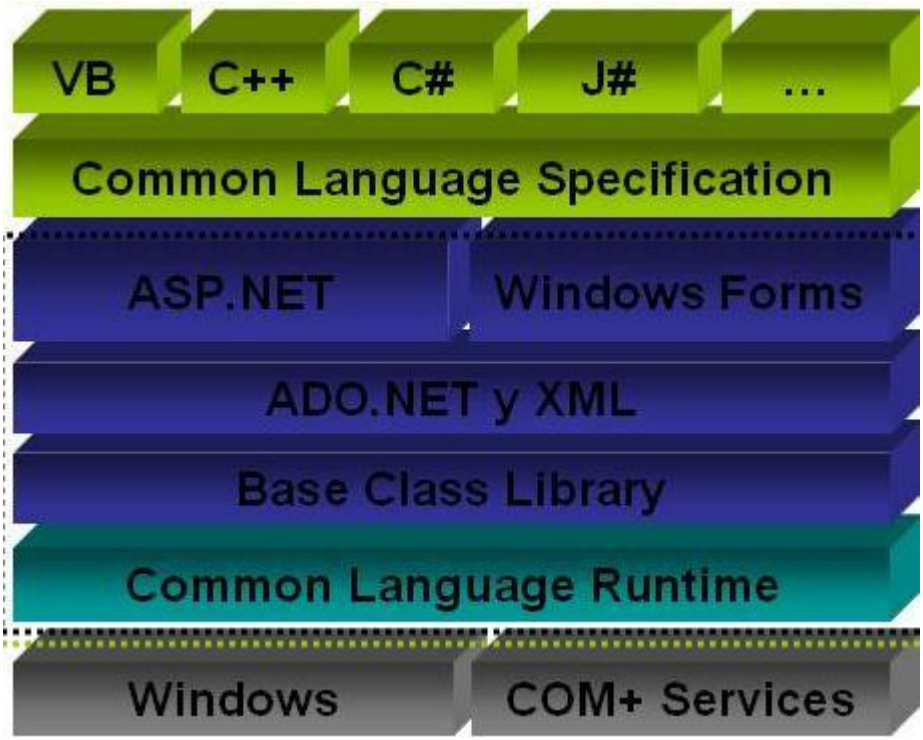
Para el tratamiento de peticiones consta de un motor PHP programado para ser nexo de unión entre el servidor, limitado a servidores Apache, y la base de datos, dando la respuesta a la petición cliente. El complemento para el replicado y el equilibrado de carga del controlador nativo de MySQL está implementado como extensión PHP. Es registrado en el arranque del intérprete, fase de inicialización de los módulos del motor PHP, como complemento del controlador para reemplazar los métodos seleccionados.

Durante la ejecución, se inspeccionan las consultas enviadas a la base de datos MySQL. Dicho complemento maneja el acceso a la conexión de la base de datos a los servidores. Fuera de este lenguaje se limita su usabilidad y portabilidad, sin separar aplicación y presentación. Respecto a Java es más rápido en servidores pequeños, pero no es totalmente compatible ni portable.

2.1.3 .NET

.NET es un marco de trabajo de Microsoft para la creación de una plataforma de desarrollo rápido. Se caracteriza por la transparencia de redes, la independencia en la plataforma hardware, interoperabilidad con los entornos y los soporte para las aplicaciones web y remotas. Su programación es fácil e intuitiva, basándose en diferentes módulos proporcionados. Los principales componentes son el conjunto de lenguajes de programación, soportando más de 20 lenguajes, la biblioteca de clases base (BLC) y el entorno común de ejecución para lenguajes (CLR). El CLR provee servicios de compilación, gestión de memoria, gestión de permisos y excepciones, depuración y seguridad. La compilación se basa en MSIL (Microsoft Intermediate

Language), lenguaje intermedio y universal en el que los demás lenguajes generan dicho código para que sea ejecutado por CLR (Common Language Runtime).



Cuenta con componentes para complementar las aplicaciones. Entre ellos cabe destacar:

- ASP.Net: HTML que contiene script procesados por un servidor antes de que sean enviados al usuario. Se puede usar la combinación con el lenguaje extensible de marcas (XML).
- ADO.Net: Acceso a los datos orientado a objetos, con servicio en el acceso a la base de datos y persistencia en la misma.
- WebForms: Formularios utilizados para aplicaciones ASP.Net.
- XML Service: Componente software que comunica con otras aplicaciones codificando mensajes en XML y enviándolos a través de protocolos estándares como HTTP.
- SOAP: Simple Object Acces Protocol en inglés, es un protocolo estándar que define la comunicación de procesos a través de XML.

Las implementaciones en .NET sólo se pueden comprar en Microsoft, por lo que delimita su uso.

La aplicación web tiene una arquitectura de servidor y cliente dinámicos. Para el desarrollo de aplicaciones web con este tipo de arquitectura hay que tener en cuenta tres tipos de tecnologías, las tecnologías cliente, las tecnologías servidor y las tecnologías de base de datos.

Las tecnologías utilizadas para la creación de páginas web son HTML, CSS y JavaScript. HTML es un lenguaje de etiquetas, que describe los distintos contenidos del documento. CSS es un lenguaje para dar formato y estilo a las etiquetas HTML y JavaScript es el lenguaje de programación del lado cliente.

Dentro de las tecnologías clientes existen multitud de librerías y frameworks de desarrollo. Algunas de las librerías más populares de JavaScript son JQuery y UnderScore, mientras que

sus Frameworks más populares son Angular.js y BackOne.js. En cuanto a los frameworks de CSS podemos encontrar Bootstrap , que posee infinidad de estilos de página web.

Todas estas tecnologías tienen que seguir los estándares web w3.org para garantizar la compatibilidad con los navegadores.

2.1.4 COMPUTACIÓN EN LA NUBE (CLOUD COMPUTING)

En este sub-apartado se aborda el paradigma de "Cloud Computing" o computación en nube, y en particular se describe la plataforma Google App Engine (GAE)[vi] basada en el siguiente paradigma:

La computación en nube, del inglés "Cloud Computing", es un paradigma que permite ofrecer servicios de computación a través de Internet, de forma transparente para el cliente, en términos de infraestructura. En este tipo de computación todo lo que puede ofrecer un sistema informático se hace como servicio desde la nube de Internet, de modo que los clientes y desarrolladores puedan acceder a ellos sin necesidad de invertir en potentes infraestructuras informáticas propias y abstrayéndose de la gestión de los recursos físicos empleados.

Ilustración 2-5 Plataforma en la Nube de Google[vii].

La mayoría de los proveedores de "Cloud Computing"[viii] ofrecen servicios escalables dinámicamente que hacen uso de recursos virtuales y son facturados siguiendo un modelo clásico de suministro de servicios: el usuario paga por la cantidad y calidad de los servicios disfrutados. El paradigma "Cloud Computing" surge como una evolución lógica de los servicios de computación y almacenamiento de datos. De la misma manera que cien años atrás el consumo de energía eléctrica pasó a ser cubierto por las grandes compañías de abastecimiento de la red eléctrica, los proveedores de servicios "Cloud Computing" representan en cierta forma algo similar, pero relativo al proceso de aplicaciones y datos. En este contexto durante las últimas décadas se ha evolucionado desde los servidores en CPDs propios hasta el paradigma "Cloud Computing", pasando por la externalización o "hosting" de servidores.

Durante dicha evolución, es necesario mencionar también como alternativa al "Cloud Computing" el paradigma "Grid Computing", que propugna la utilización de recursos heterogéneos no sujetos a un control central de forma coordinada. De hecho, ambos movimientos cuentan con igual número de seguidores y detractores, si bien ciertos autores consideran que el "Cloud Computing" no es más que un tipo de "Grid Computing", dado que se trata de un paradigma que surgió con cierta posterioridad.

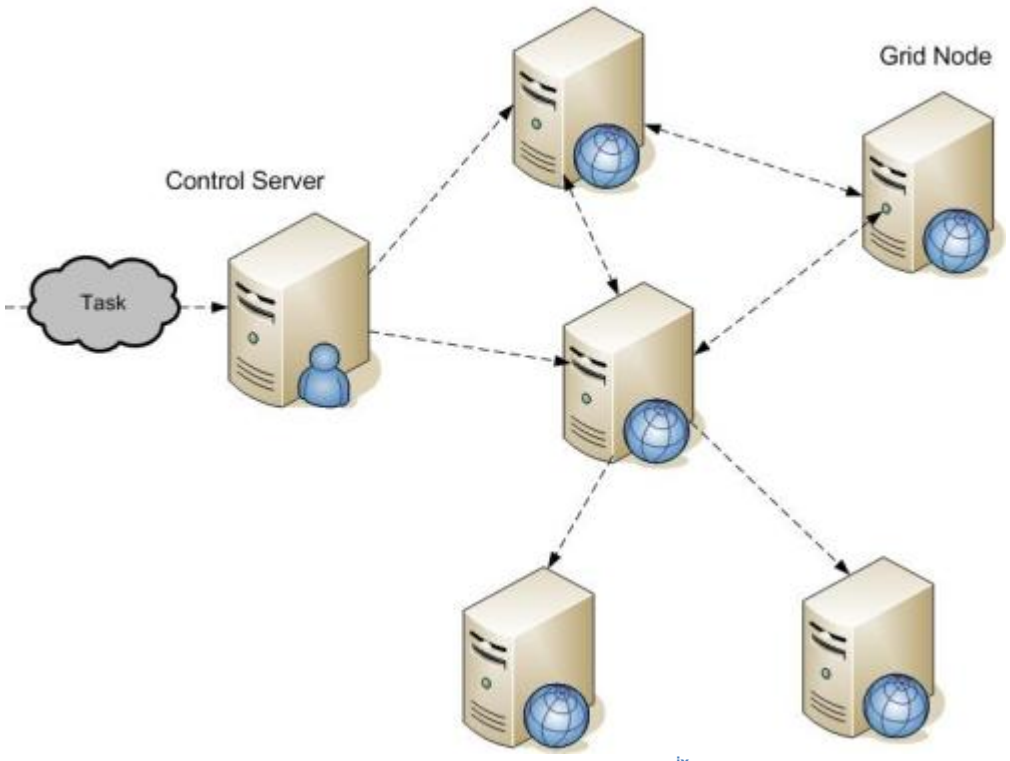


Ilustración 2-6 Grid Computing[ix]

Dentro de un modelo de "Cloud Computing" se distinguen varias capas entre el cliente y la infraestructura.

Cliente: componentes hardware y/o sistema software que hacen uso de los servicios ofrecidos por la nube, o en caso estricto han sido diseñados en exclusiva con ese fin y no tienen utilidad en otro contexto. Por ejemplo: clientes móviles (terminales móviles), clientes ligeros (sistemas operativos basados en "Cloud Computing") y clientes pesados (navegadores web).

Aplicación: actúa como enlace entre el cliente y la plataforma, ofreciendo los servicios al usuario y gestionando los recursos empleados en la plataforma. En muchos casos se trata de aplicaciones que no requieren instalación ni ejecución en el lado del cliente, evitando las tareas relacionadas con la instalación, mantenimiento y soporte del software.

Plataforma: ofrece una plataforma de computación y/o solución de pila como un servicio. Facilita el despliegue de aplicaciones sin el coste y la complejidad de comprar y gestionar el hardware subyacente y las capas de software.

Infraestructura: recursos físicos empleados por la plataforma y con ubicación desconocida para el cliente. En muchos casos se encuentran agrupados en grandes centros de computación.

Como ejemplos de servicios de "Cloud Computing" públicos destacan: Google App Engine, Amazon EC2 y Windows Azure, que proveen aplicaciones comunes en línea accesibles desde distintos dispositivos, mientras el software y los datos se almacenan en los servidores. Todos ellos se basan en el mismo paradigma, pese a que cada uno posee sus particularidades y en algunos

casos existen diferencias notables entre ellos. Por ejemplo, los dos servicios más populares, Google App Engine y Amazon EC2, difieren en el modo en el que las aplicaciones desarrolladas son migradas a la infraestructura física. GAE realiza esta migración de forma automática, en función de las necesidades dinámicas de las ejecuciones de la aplicación, mientras que con Amazon EC2 es el mismo desarrollador el encargado del mapeo físico de sus aplicaciones. Esto último permite un mayor grado de control, pero al mismo tiempo implica una mayor carga de trabajo.

Google App Engine (GAE) es una plataforma concebida para desarrollar, alojar y ejecutar aplicaciones web sobre la infraestructura Google. La plataforma hace uso del paradigma "Cloud Computing", mediante la virtualización de aplicaciones a través los numerosos servidores de los centros de datos de Google, diseminados por todo el mundo. La infraestructura Google es totalmente transparente para el cliente de los servicios "Cloud Computing", quien se despreocupa de la gestión de los recursos utilizados, mientras que el usuario desarrollador por su parte es capaz de crear, mantener y actualizar sus aplicaciones.

Ilustración 2-7 Google App Engine

Se entiende como usuario desarrollador al programador de aplicaciones sobre la plataforma GAE, para que más tarde éstas sean ejecutadas por los clientes de los servicios. Al contrario que plataformas como Amazon EC2, que virtualizan a nivel de imágenes de máquinas virtuales, GAE ofrece su infraestructura para contener aplicaciones exclusivamente. GAE ejecuta de forma eficiente aplicaciones escalables gracias a una entidad llamada Balanceador de Carga, que se encarga de asignar más o menos recursos (servidores) a cada una de ellas. El esquema del funcionamiento de Google App Engine es el siguiente: clientes acceden a las aplicaciones, que son implementadas por un número de servidores determinado por el Balanceador de Carga, en función de los recursos necesarios y/o disponibles. Por su parte, los servidores se sirven de la API que les es proporcionada por Google y al mismo tiempo también pueden hacer uso de otros servicios de la infraestructura Google como la BigTable o las cuentas de usuario (Google Accounts). La arquitectura de la plataforma está hecha para diferentes tipos de cliente o peticiones entrantes, como por ejemplo navegadores o dispositivos móviles.

2.1.5 BLUEMIX

BlueMix[x] es la implementación de la Arquitectura Abierta en la Nube de IBM basada en Cloud Foundry, que permite crear, desplegar y administrar rápidamente sus aplicaciones en Cloud. Dado que BlueMix está basado en Cloud Foundry, puede aprovechar el ecosistema de frameworks y servicios de tiempo de ejecución en crecimiento.

BlueMix proporciona aquellos servicios de nivel básico y empresarial que necesitan las organizaciones para que sus aplicaciones en la nube estén listas y disponibles para sus clientes, cuando las necesiten y donde lo necesiten. Al estar basado en una tecnología abierta, BlueMix proporciona la flexibilidad necesaria para integrar el marco de desarrollo y aquellos servicios que se adapten a las necesidades de cada empresa.

2.1.6 DOCKER

Docker[xi] es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux. Docker utiliza características de aislamiento de recursos del kernel de Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

El soporte del kernel de Linux para los espacios de nombres aísla de vista, en su mayoría, una aplicación del entorno operativo, incluyendo árboles de proceso, red, ID de usuario y sistemas de archivos montados, mientras que los cgroups del kernel proporcionan aislamiento de recursos, incluyendo la CPU, la memoria, el bloque de E/S y de la red.

Resumiendo: Docker es una herramienta que puede empaquetar una aplicación y sus dependencias en un contenedor virtual que se puede ejecutar en cualquier servidor Linux.

2.1.7 BASES DE DATOS NO-SQL

Cuándo pensamos en bases de datos relacionales, a nuestra mente suelen acudir los mismos nombres. En la parte comercial tenemos Oracle y Microsoft SQL Server. Del lado del software libre, tenemos opciones como Postgre SQL o MySQL. Aunque cada una tiene sus peculiaridades, para un desarrollador no es difícil elegir entre un sistema y otro. Al final todo son tablas, columnas, claves primarias, y sobre todo, consultas SQL. La decisión de cuál elegir, se basará en sus características y precio.

Si hablamos de **bases de datos NoSQL**[xii], la cosa se complica. A día de hoy existen unos 150 sistemas de bases de datos NoSQL. Elegir uno de ellos puede ser muy difícil, ya que ninguno ha obtenido todavía la fama que sí han conseguido las bases de datos relacionales.

Aunque hay varias aproximaciones diferentes para clasificar las bases de datos NoSQL (Teorema CAP, basándonos en el modelo de datos, etc.), en general, se considera que existen cuatro tipos diferentes: orientadas a documentos, orientadas a columnas, de clave-valor y en grafo. Así que

veamos en qué consisten estos sistemas, para que podamos elegir la opción que mejor se adapte a nuestras necesidades**.**

Orientadas a documentos:

Son aquéllas que gestionan datos semi-estructurados, es decir, documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON.

Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales.

En esta categoría encontramos:

MongoDB: probablemente la base de datos NoSQL más famosa del momento. En octubre del año pasado, MongoDB conseguía 150 millones de dólares en financiación, convirtiéndose en una da las startups más prometedoras. Algunas compañías que actualmente utilizan MongoDB son Foursquare o eBay.

CouchDB: es la base de datos orientada a documentos de Apache. Una de sus interesantes características es que los datos son accesibles a través de una API Rest. Este sistema es utilizado por compañías como Credit Suisse y la BBC.

Orientadas a columnas:

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros.

En esta categoría encontramos:

Cassandra: incluida en esta sección, aunque en realidad sigue un modelo híbrido entre orientada a columnas y clave-valor. Es utilizada por Facebook y Twitter (aunque dejaron de usarla para almacenar tweets).

HBase. Escrita en Java y mantenida por el Projecto Hadoop de Apache, se utiliza para procesar grandes cantidades de datos. La utilizan Facebook, Twitter o Yahoo.

De clave valor:

Éstas son las más sencillas de entender. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor.

En esta categoría encontramos:

DynamoDB: desarrollada por Amazon, es una opción de almacenaje que podemos usar desde los Amazon Web Services. La utilizan el Washington Post y Scopely.

Redis: desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow (a modo de caché).

En grafo:

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales.

En esta categoría encontramos:

Infinite Graph: escrita en Java y C++ por la compañía Objectivity. Tiene dos modelos de licenciamiento: uno gratuito y otro de pago.

Neo4j: base de datos de código abierto, escrita en Java por la compañía Neo Technology. Utilizada por compañías como HP, Infojobs o Cisco.

2.1.8 BIG DATA

Las nuevas aplicaciones se enfrentan[xiii] a nuevos desafíos que les permitan analizar, descubrir y entender más allá de lo que sus herramientas tradicionales reportan sobre su información. Existe una tendencia en el avance de la tecnología que ha abierto las puertas hacia un nuevo enfoque de entendimiento y toma de decisiones, la cual es utilizada para describir enormes cantidades de datos (estructurados, no estructurados y semi-estructurados) que tomaría demasiado tiempo y sería muy costoso cargarlos a una base de datos relacional para su análisis. De tal manera que, el concepto de Big Data se aplica para toda aquella información que no puede ser procesada o analizada utilizando procesos o herramientas tradicionales. Sin embargo, Big Data no se refiere a alguna cantidad específica, y es usualmente utilizado cuando se habla en términos de petabytes y exabytes de datos.

Además del gran volumen de información, ésta existe en una gran variedad de datos, que pueden ser representados de diversas maneras en todo el mundo, por ejemplo de dispositivos móviles, audio, video, sistemas GPS, incontables sensores digitales en equipos industriales, automóviles, medidores eléctricos, veletas, anemómetros, etc., los cuales pueden medir y comunicar el posicionamiento, movimiento, vibración, temperatura, humedad y hasta los cambios químicos que sufre el aire, de tal forma que las aplicaciones que analizan estos datos requieren que la velocidad de respuesta sea lo suficiente rápida para lograr obtener la información correcta en el momento preciso. Éstas son las características principales de una oportunidad para Big Data.

Es importante entender que las bases de datos convencionales son una parte importante y relevante para una solución analítica. De hecho, se vuelve mucho más vital cuando se usa en conjunto con la plataforma de Big Data.

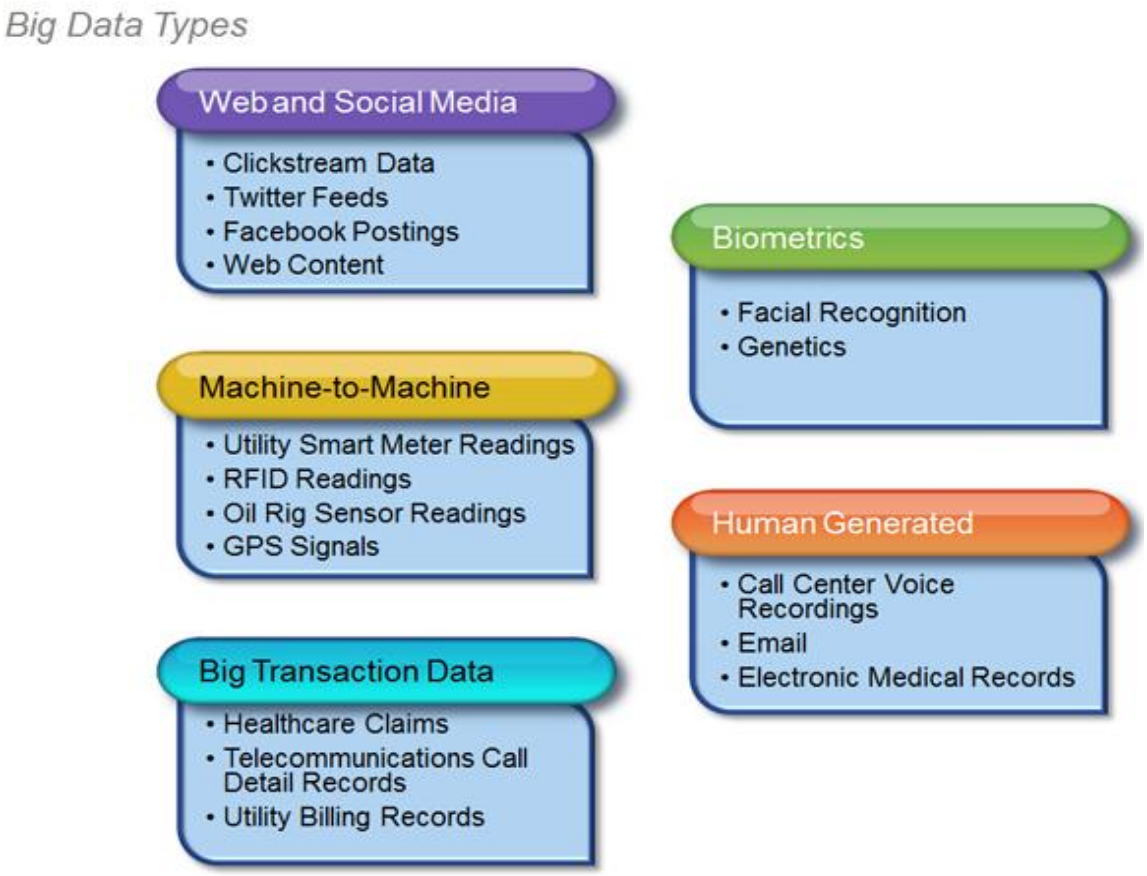


Ilustración 2-8 Tipos de datos de Big Data

1.- Web and Social Media: Incluye contenido web e información que es obtenida de las redes sociales como Facebook, Twitter, LinkedIn, etc,

2.- Machine-to-Machine (M2M)*:* M2M se refiere a las tecnologías que permiten conectarse a otros dispositivos. M2M utiliza dispositivos como sensores o medidores que capturan algún evento en particular (velocidad, temperatura, presión, variables meteorológicas, variables químicas como la salinidad, etc.) los cuales transmiten a través de redes alámbricas, inalámbricas o híbridas, a otras aplicaciones que traducen estos eventos en información significativa.

3.- Big Transaction Data: Incluye registros de facturación, en telecomunicaciones registros detallados de las llamadas (CDR), etc. Estos datos transaccionales están disponibles en formatos tanto semi-estructurados como no estructurados.

4.- Biometrics: Información biométrica en la que se incluyen huellas digitales, escaneo de la retina, reconocimiento facial, genética, etc. En el área de seguridad e inteligencia, los datos biométricos han sido información importante para las agencias de investigación.

5.- Human Generated: Las personas generamos diversas cantidades de datos como la información que guarda un call center al establecer una llamada telefónica, notas de voz, correos electrónicos, documentos electrónicos, estudios médicos, etc.

2.1.9 VERTX

Vert.x es un framework para el desarrollo de aplicación orientada a eventos que se ejecuta en la máquina virtual de Java.

Sus principales características son:

- Polyglot: las aplicaciones pueden ser escritas en Java, JavaScript, Groovy, Ruby o Python.
- Modelo de concurrencia simple. Todo el código que se escribe es single-threaded.
- Modelo de programación asíncrona simple: para escribir aplicaciones escalables y no bloqueantes.
- Distributed Event Bus: que se extiende por el cliente y el servidor. El bus de eventos incluso llega a la capa JavaScript en el navegador permitiendo crear aplicaciones Real Time.
- Sistema modular out-of-the-box: incluyendo un web-server, persistencia (sobre MongoDB), colas,…
- Soporte para:
    - HTTP/HTTPS
    - TCP/SSL
    - WebSockets
    - SockJS
    - Open Source
    - Embebible: Vert.x puede correr como un Server o embebido en nuestra aplicación

2.1.10 NODE.JS

Node.js[xiv] es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motorV8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla.

Node.js es similar en su propósito a Twisted o Tornado de Python, Perl Object Environment de Perl, React de PHP, libevent o libev de C, EventMachine de Ruby, vibe.d de D y de Java existe Apache MINA, Netty, Akka, Vert.x, Grizzly o Xsocket. Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Node.js implementa algunas especificaciones de CommonJS.5Node.js incluye un entorno REPL para depuración interactiva.

2.1.11 JAVASCRIPT

JavaScript[xv] es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript podemos crear diferentes efectos e interactuar con nuestros usuarios. JavaScript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado.

Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, es un lenguaje que utiliza Windows y sistemas X-Windows; gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

Es necesario resaltar que hay dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, éste es el JavaScript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un JavaScript que se ejecuta en el servidor, es más reciente y se denomina LiveWire JavaScript.

JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Chrome, Opera, Mozilla Firefox, entre otros.

Con el surgimiento de lenguajes como PHP del lado del servidor y JavaScript del lado del cliente, surgió Ajax en acrónimo de (Asynchronous JavaScript And XML). El mismo es una técnica para crear aplicaciones web interactivas. Este lenguaje combina varias tecnologías:

- HTML y Hojas de Estilos CSS para generar estilos.
- Implementaciones ECMAScript, uno de ellos es el lenguaje JavaScript.
- XMLHttpRequest es una de las funciones más importantes que incluye, que permite intercambiar datos asincrónicamente con el servidor web, puede ser mediante PHP, ASP, entre otros.

2.1.12 SOAP[xvi]

En el núcleo de los servicios Web se encuentra el protocolo simple de acceso a datos SOAP, que proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido gran atención debido a que facilita una comunicación del estilo RPC entre un cliente y un servidor remoto. Pero existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC de Sun, DCE de Microsoft, RMI de Java y ORPC de CORBA. ¿Por qué se presta tanta atención a SOAP?

Una de las razones principales es que SOAP ha recibido un increíble apoyo por parte de la industria. SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores Compañías que soportan SOAP son Microsoft, IBM, SUN Microsystems, SAP y Ariba.

Algunas de las Ventajas de SOAP son:

- No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el ultimo y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- No está atado a ninguna infraestructura de objeto distribuido. La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.
- Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y, como ya se ha mencionado, SOAP no define un medio de transporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- Permite la interoperabilidad entre múltiples entornos: SOAP se desarrolla sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

2.1.13 API REST

REST deriva de "Representational State Transfer", es decir, transferencia de representación de estado. Un servicio REST no tiene estado (es stateless), lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Esto es, que no se puede llamar a un servicio REST y pasarle unos datos (p. ej. un usuario y una contraseña) y esperar que “nos recuerde” en la siguiente petición. De ahí el nombre: el estado lo mantiene el cliente y por lo tanto es el cliente quien debe pasar el estado en cada llamada. Si quiero que un servicio REST me recuerde, debo pasarle quien soy en cada llamada. Eso puede ser un usuario y una contraseña, un token o cualquier otro tipo de credenciales, pero debe pasarlas en cada llamada. Y lo mismo aplica para el resto de información.

REST[xvii] no es una tecnología, ni siquiera una arquitectura, sino una familia de arquitecturas. Cualquier arquitectura de servicios distribuidos que cumpla con una serie de requisitos se puede considerar como una arquitectura REST. Estos requisitos o propiedades son los siguientes:

- No se publican servicios RPC. En arquitecturas REST, los servicios no publican un conjunto arbitrario de métodos u operaciones. Por ejemplo, en REST no podemos publicar una interfaz "IGestionEmpleados" con métodos "addEmpleado", "removeEmpleado" o "buscarEmpleadosEnEdadDeJubilacion".
- En REST lo que se publica son *recursos*. Un recurso se puede considerar como una entidad que representa un concepto de negocio que puede ser accedido públicamente. Un ejemplo de recurso sería simplemente "EmpleadosDeLaEmpresa" y otro podría ser "Empleado número 33"
- Cada recurso, como buena entidad que se precie, y de acuerdo a los principios de OO, posee un identificador único y global, que lo distingue de cualquier otro recurso, aunque ambos tuvieran exactamente los mismos datos. En el caso de "Empleado 33", éste sería diferente de "Empleado 40", aunque tuvieran el mismo nombre, sueldo, etc.
- Cada recurso posee un estado interno, que no puede ser accedido directamente desde el exterior. Lo que sí es accesible desde el exterior es una o varias representaciones de dicho estado. Por representación se entiende un formato de datos concreto usado para la transferencia de una copia del estado público del recurso entre el cliente y el servidor. La implementación del recurso decide qué información es visible o no desde el exterior, y qué representaciones de dicho estado se soportan. Una representación de "Empleado 33" podría ser un documento XML con la información accesible de éste. Otra representación sería un documento HTML y otra podría ser un JSON. No sólo podemos representar el recurso como datos estructurados, hay que echarle imaginación. Podríamos pedir, por ejemplo, una representación en formato imagen PNG del recurso, tal vez esto devolvería una foto del empleado, o un gráfico de su productividad o su huella dactilar.
- Si no podemos definir operaciones arbitrarias sobre el recurso, ¿cómo podemos operar con él? En REST todos los recursos comparten una interfaz única y constante. Todos los recursos tienen las mismas operaciones. Las operaciones nos permiten manipular el estado público del recurso. En un sistema REST típico se definen cuatro operaciones.
    - **CREATE**. En esta operación el cliente manda al servidor una petición para crear un nuevo recurso. Opcionalmente el cliente puede mandar una representación del estado inicial de este recurso. El servidor responde con el identificador global del nuevo recurso.
    - **DELETE**. En esta operación el cliente elimina un recurso del servidor. El cliente necesita saber el identificador del recurso.
    - **READ**. Con esta operación el cliente puede leer una representación del estado de un recurso, identificado con su identificador global. El cliente puede especificar qué tipos de representaciones entiende. Aquí lo que ocurre realmente es que se copia el estado del recurso en el servidor y se pega en el cliente. Ambas copias del estado no se mantienen sincronizadas. El servidor puede cambiar el estado real del recurso y el cliente, de forma independiente, puede modificar su copia local del estado del recurso.
    - **UPDATE**. Como el servidor y el cliente tienen una copia diferente del estado, el cliente puede usar esta operación para sobrescribir o grabar su copia del estado en el servidor.

De esta manera se puede actualizar el estado del recurso con las modificaciones hechas en el cliente.

- La implementación del servicio es libre de prohibir alguno de estos métodos para un recurso en concreto. También debe definir el modelo de datos que se va a publicar y que representaciones soporta. Lo que no puede hacer es inventarse operaciones de forma arbitraria. Las operaciones son siempre las mismas.
- Los distintos recursos se pueden interrelacionar y referenciar entre sí mediante sus identificadores globales.

Ventajas de un desarrollo basado en API REST:

1. Separación cliente/servidor. Al ser sistemas independientes (sólo se comunican con un lenguaje de intercambio como JSON) se pueden desarrollar proyectos autónomos, equipos autónomos. Al cliente le da igual cómo está hecha la API y al servidor le da igual qué vas a hacer con los datos que te ha proporcionado.

2. Independencia de tecnologías / lenguajes. Se puede desarrollar en cualquier tipo de tecnología o lenguaje con la que te sientas a gusto, con la que puedas acortar tus tiempos de desarrollo, o que encaje con la filosofía o necesidades del proyecto. Es indiferente que en el futuro se cambie totalmente las tecnologías con las que está implementado tu API REST, siempre y cuando se respete "el contrato", o sea, que se sigan teniendo las mismas operaciones en el API y hagan las mismas cosas que se supone que deben hacer.

3. Fiabilidad, escalabilidad, flexibilidad. Al final sólo te tienes que preocupar que el nexo cliente / servidor esté correcto. Se pueden hacer cambios en tu servidor, lenguajes, bases de datos, etc. y, mientras devuelvas los datos que toca, todo irá correctamente. Escalabilidad porque puedes crecer todo lo que necesites en cada momento. Tu API puede responder a otros tipos de operaciones o puede versionarse tanto como desees. También tu programación del lado del cliente puede crecer todo lo necesario con el tiempo, incluso, como decíamos, crear otros frontales, no sólo web, también de Apps para cualquier dispositivo.

4. Experiencia de usuario. Aunque eso depende más de cómo está hecha la parte del cliente, teóricamente el desarrollo de sitios web basados en un API puede dar mejor desempeño que uno tradicional. Cuando haces una solicitud al servidor lo que tienes como respuesta son datos planos, que requieren tiempos de transferencia menores que si esos mismos datos los recibieras mezclados con el HTML/CSS de la presentación. En este tipo de aplicaciones web no necesitas recargar la página, aunque esto no es una ventaja específica del desarrollo basado en REST, sino del uso de Ajax en general, con el que podemos conseguir aplicaciones web que se asemejan más a aplicaciones de escritorio

5. REST requiere menos recursos del servidor. Esto no es necesariamente cierto aunque en muchos casos sí se pueda deducir. Hay muchas opiniones al rededor de REST. Nosotros basamos esa afirmación en estos motivos:

- No mantener el estado, no requiere memoria, se pueden atender más peticiones.
- No requiere escribir el HTML, por lo tanto tienes menos procesamiento en el servidor.

2.1.14 BOOTSTRAP

Bootstrap[xviii] es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT en el año 2011 y su desarrollo continúa en un repositorio de GitHub.

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

Desde la aparición de Bootstrap 3, el framework se ha vuelto bastante más compatible con desarrollo web responsive, entre otras características se han reforzado las siguientes:

- Soporte bastante bueno (casi completo) con HTML5 y CSS3, permitiendo ser usado de forma muy flexible para desarrollo web con unos excelentes resultados.
- Se ha añadido un sistema GRID que permite diseñar usando un GRID de 12 columnas donde se debe plasmar el contenido, con esto podemos desarrollar responsive de forma mucho más fácil e intuitiva.
- Boostrap 3 establece Media Queries para 4 tamaños de dispositivos diferentes, variando dependiendo del tamaño de su pantalla, estas Media Queries permiten desarrollar para dispositivos móviles y tablets de forma mucho más fácil.
- Boostrap 3 también permite insertar imágenes responsive, es decir, con sólo insertar la imagen con la clase “img-responsive” las imágenes se adaptaran al tamaño.

Todas estas características hacen que Boostrap sea una excelente opción para desarrollar webs y aplicaciones web totalmente adaptables a cualquier tipo de dispositivo.

Boostrap es compatible con la mayoría de navegadores web del mercado, y más desde la versión 3, actualmente es totalmente compatible con los siguientes navegadores:

- Google Chrome (en todas las plataformas).
- Safari (tanto en iOS como en Mac).
- Mozilla Firefox (en Mac y en Windows).
- Internet Explorer (en Windows y Windows Phone).
- Opera (en Windows y Mac).

2.1.15 JSF

JavaServer Faces (JSF)[xix] es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL (acrónimo de XML-based User-interface Language, lenguaje basado en XML para la interfaz de usuario)

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

Los siguientes objetivos representan el foco de desarrollo de JSF:

1. Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.
2. Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.
3. Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.
4. Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
5. Especificar un modelo para la internacionalización y localización de la interfaz de usuario.
6. Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

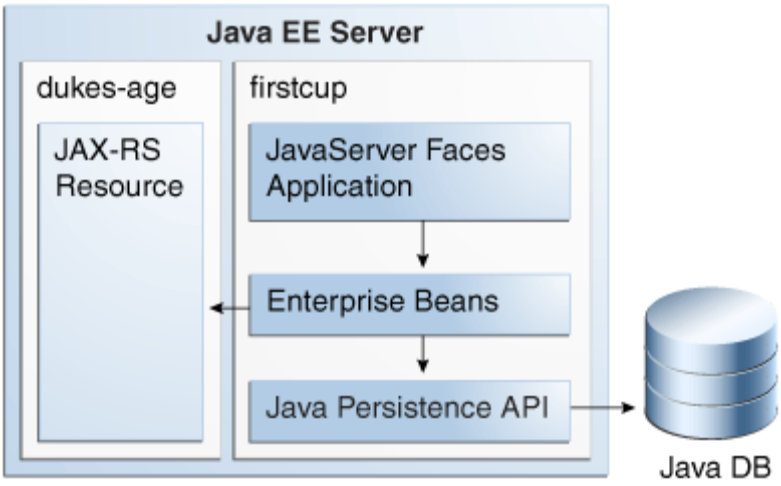


Ilustración 2-9 Arquitectura de una Aplicación JSF simple

2.1.16 JPA

Java Persistence API[xx], más conocida por sus siglas **JPA**, es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).
Persistencia en este contexto cubre tres áreas:

- La API en sí misma, definida en el paquete javax.persistence
- El lenguaje de consulta Java Persistence Query Language (JPQL).
- Metadatos objeto/relacional.

El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional), como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs).

Una entidad de persistencia (entity) es una clase de Java ligera, cuyo estado es persistido de manera asociada a una tabla en una base de datos relacional. Las instancias de estas entidades corresponden a un registro (conjunto de datos representados en una fila) en la tabla. Normalmente las entidades están relacionadas a otras entidades, y estas relaciones son expresadas a través de meta datos objeto/relacional. Los meta datos del objeto/relacional pueden ser especificados directamente en el fichero de la clase, usando las anotaciones de Java (annotations), o en un documento descriptivo XML, el cual es distribuido junto con la aplicación.

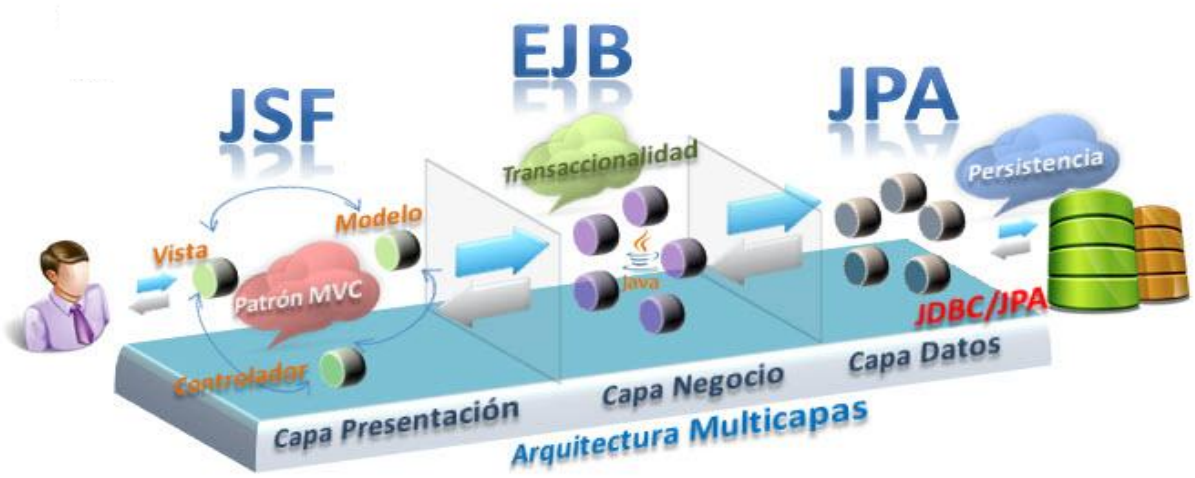


Ilustración 2-10 JPA[xxi]

2.2 METODOLOGÍA

Una metodología de desarrollo de software se refiere a un marco de trabajo que es usado para estructurar, planificar y controlar el proceso de desarrollo de los sistemas de información.

Existen numerosas metodologías de desarrollo, de las cuales las más extendidas y usadas son:

1. Metodología en cascada.
2. Metodología en V.
3. Metodología en espiral.
4. Metodologías ágiles.

Veamos cada una de ellas en detalle:

2.2.1 METODOLOGÍA EN CASCADA

La metodología de desarrollo en cascada es un proceso secuencial en el que cada fase del desarrollo es vista hacia abajo (como una cascada), de tal forma que el inicio de la siguiente etapa debe esperar a la finalización de la etapa previa.

La definición de este modelo fue propuesta por el Dr. Winston W. Royce en 1970, pero no fue hasta los años 80 cuando el modelo se refinó, primeramente por Boehm y luego por Sommerville para dar lugar a lo que hoy se conoce como modelo en cascada.

Las etapas del modelo en cascada suelen variar, pero en general se dividen en:

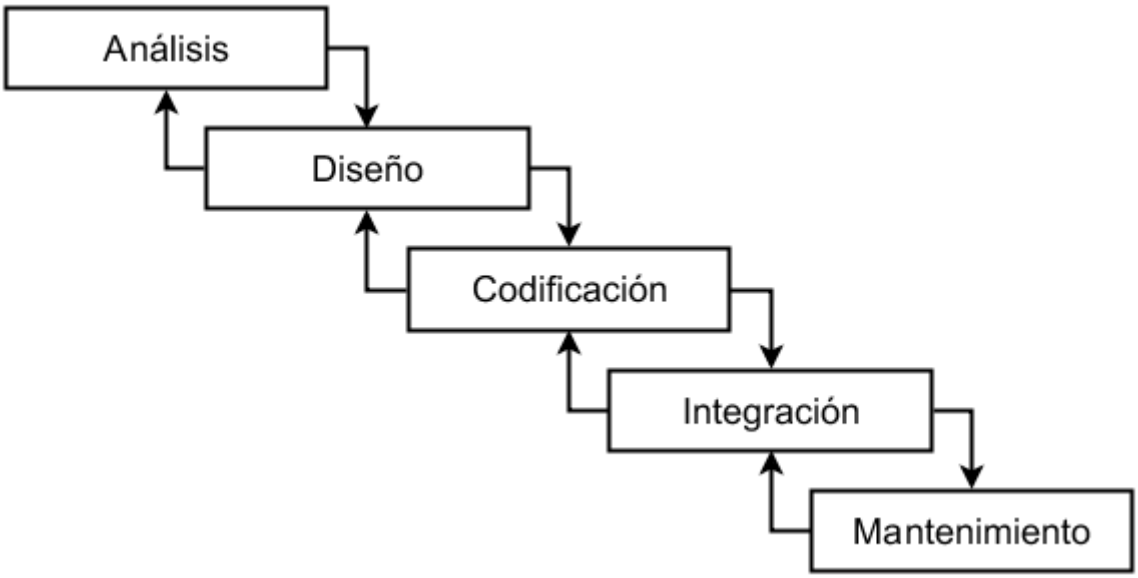


Ilustración 2-11 Etapas del Modelo en Cascada

1. Análisis de requisitos
2. Diseño preliminar
3. Diseño detallado
4. Codificación
5. Pruebas
6. Verificación
7. Mantenimiento.

La implementación de este modelo en los sistemas de información de la vida real suele conducir al fracaso, debido a que los proyectos raramente siguen una secuencia lineal. Otro problema que se puede achacar, es que cualquier error de diseño que se haya detectado en la etapa de pruebas, conduce al rediseño y nueva programación, por lo que los costes del proyecto y fechas de entrega pueden estar seriamente afectados.

2.2.2 METODOLOGÍA EN V

Está basado en el modelo en cascada, se describen las actividades y los resultados del desarrollo del producto. En el lado izquierdo de la "V" se representa las necesidades y las especificaciones del sistema. En el otro lado se representa la integración y validación del sistema. Entre los dos lados forman la verificación y validación de la aplicación.

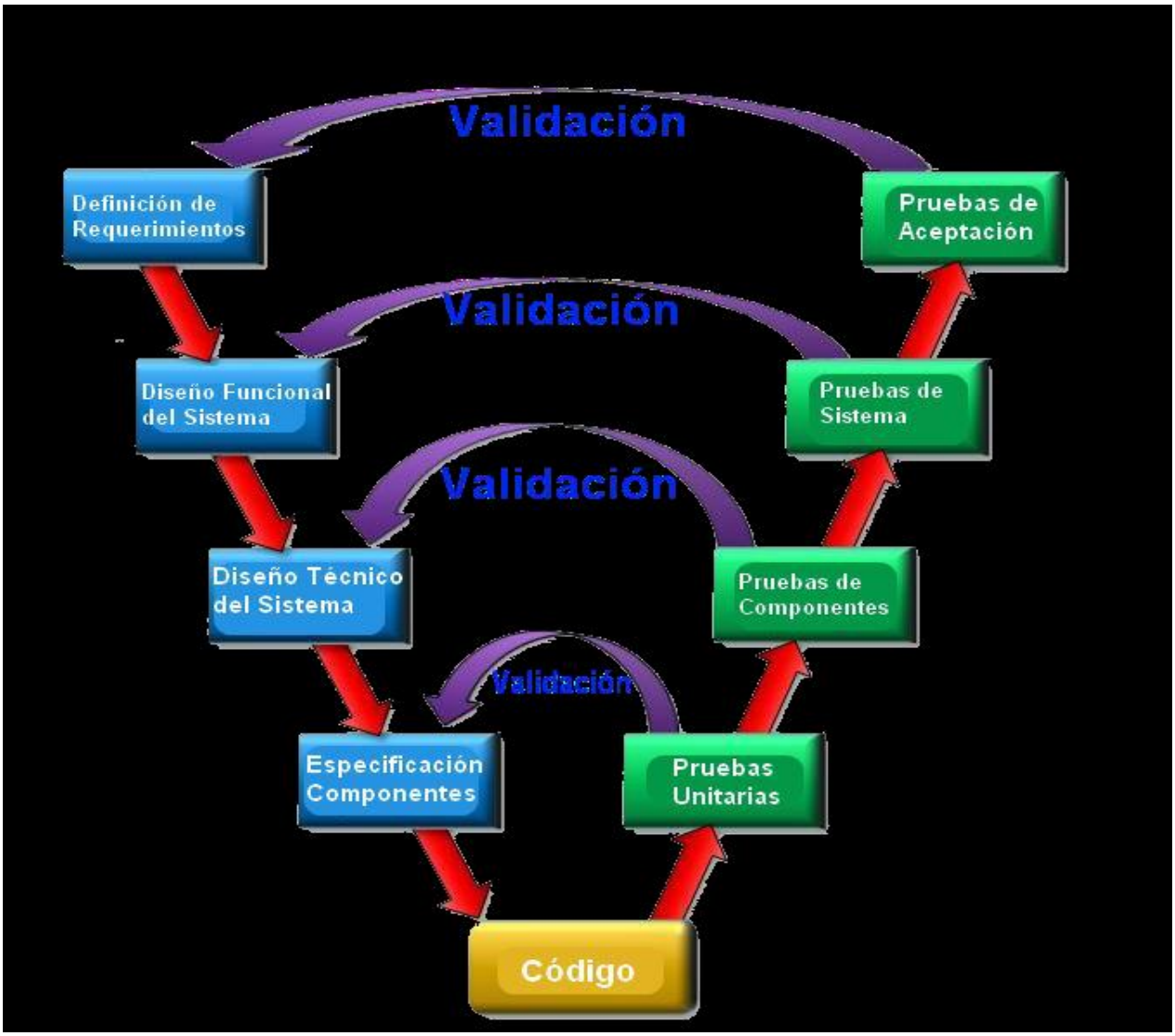


Ilustración 2-12 Metodología en V

2.2.3 METODOLOGÍA EN ESPIRAL

Las actividades se estructuran en una espiral en la que cada iteración representa el conjunto de actividades, las cuales no tienen prioridad.

Cada nueva actividad se elige en función del análisis de riesgo en el anterior bucle.

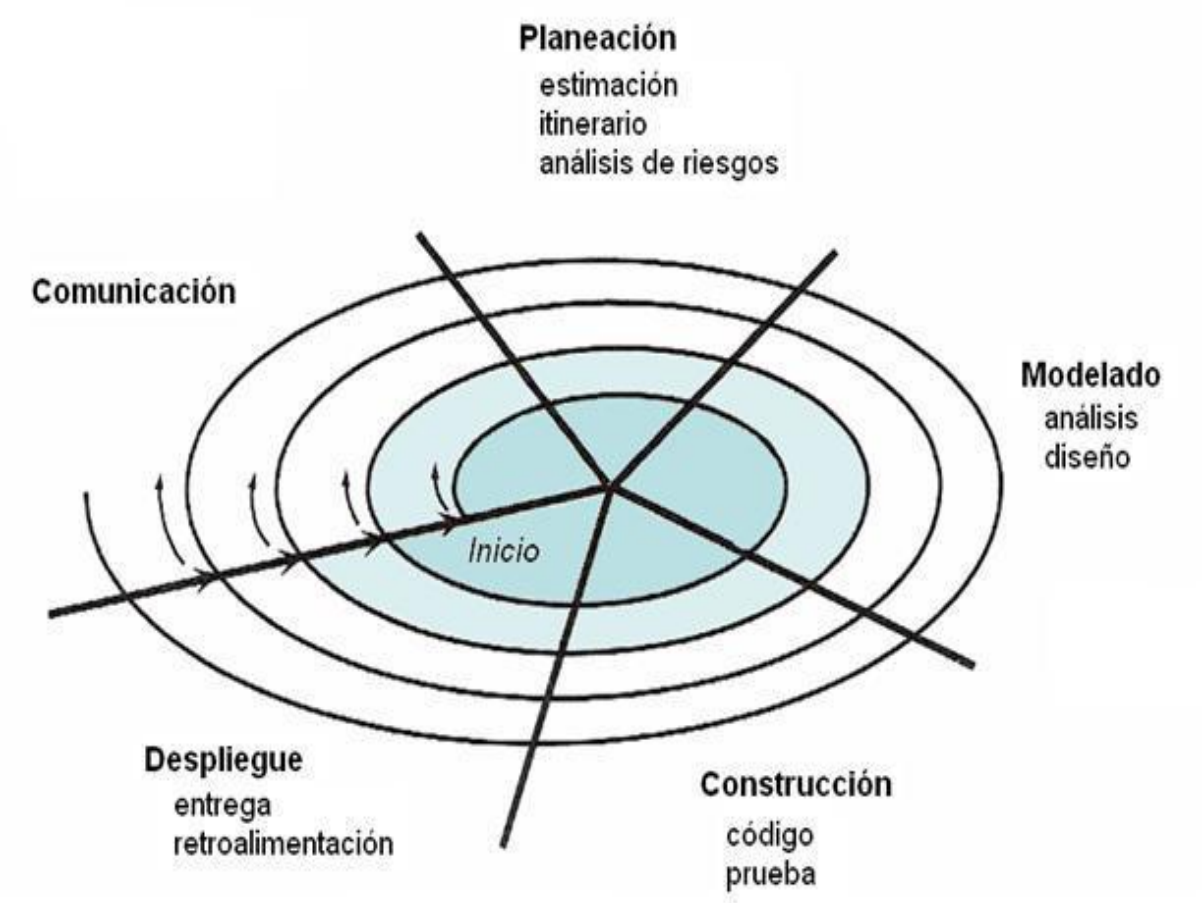


Ilustración 2-13 Metodología en Espiral

Las metodologías se pueden dividir por su enfoque hacia el análisis y el diseño, como las metodologías estructuradas, las metodologías orientadas a objetos y las metodologías con enfoque al control del proyecto. Este último enfoque controla todo el desarrollo del proyecto, especialmente su planificación. Existen dos tipos dentro de este grupo: las tradicionales y las ágiles.

2.2.4 METODOLOGÍA TRADICIONAL

Metodología guiada por una fuerte planificación durante todo el proceso de desarrollo. Realiza el análisis y el diseño antes de la construcción de todo el sistema, pretendiendo prever todo de antemano. Los ejemplos más destacados son Métrica V.3, siendo el Ministerio de Hacienda y Administraciones Públicas como fuente de su propiedad intelectual, y RUP (Rational Unified Process), producto de IBM caracterizado por ser iterativo e incremental, centrado en la arquitectura.

Dividen el ciclo de vida para el desarrollo en cuatro fases:

- Inicio: determinar la visión del proyecto y definir lo que se desea, complementando así el modelado de negocio y definiendo los requisitos para estimar los costes y el tiempo en el desarrollo del sistema.
- Elaboración: determinar la arquitectura óptima trasladando los requisitos analizados a un sistema automatizado.
- Construcción: desarrollo de la capacidad operacional implementando el software ajustado a la arquitectura diseñada y realizando pruebas para asegurar su correcto funcionamiento.
- Transmisión: se obtiene el producto definido y terminado listo para su distribución.

Tanto Métrica v3 como RUP se pueden basar en seis principios clave:

- Adaptación del proceso: el proceso debe adaptarse a las características de la organización para la que se está desarrollando el software.
- Balancear prioridades: debe encontrarse un equilibrio que satisfaga a los inversores.
- Colaboración entre equipo: debe de haber comunicación fluida para cualquier problema que se ocasione.
- Demostrar valor iterativamente: de forma interna se entrega el producto en diversas etapas iteradas del proyecto. En cada una se evalúa el producto con la claridad y la estabilidad del mismo.
- Elevar el nivel de abstracción: usar los conceptos reutilizables.
- Enfocarse en la calidad: verificable en cada aspecto del proyecto.

2.2.5 METODOLOGÍA ÁGIL[xxii]

La metodología de desarrollo ágil se refiere a los métodos basados en el desarrollo iterativo e incremental que implicará equipos multifuncionales, es decir, que no hay sólo una persona con conocimiento exclusivo para hacer algo. También implica la auto-organización, referido a que en la mayoría de los proyectos ágiles no hay, por ejemplo, un único jefe de proyecto responsable de asignar tareas.

La historia de la metodología ágil obtuvo su reconocimiento en el año 2001, cuando destacados y conocidos miembros de la ingeniería del software escribían en Utah el Manifiesto.

El objetivo consistía en establecer valores y principios, para permitir a los equipos el desarrollo de software rápidamente y poder responder a cambios que pudiesen surgir a lo largo del proyecto.

En un proyecto ágil el diseño, el desarrollo y las pruebas se realizan de manera continua. De esta forma, se lleva la iteración al extremo:

- Se busca dividir las tareas del proyecto en incrementos con una planificación mínima y de corta duración (entre 1 y 4 semanas).

- Cada iteración suele concluir con un prototipo operativo. Al final de cada iteración, se obtiene un producto entregable que es revisado junto con el cliente, posibilitando la aparición de nuevos requisitos o perfeccionando los existentes.

La reducción de los riesgos globales o la capacidad de adaptación rápida a los cambios que pudieran surgir en el proyecto son unas de las ventajas que poseen las metodologías ágiles.

Los cuatro valores de la metodología ágil son[xxiii]:

1. Valorar más a los individuos y su interacción que a los procesos y las herramientas.
2. Valorar más el software que funciona que la documentación exhaustiva.
3. Valorar más la colaboración con el cliente que la negociación contractual.
4. Valorar más la respuesta al cambio que el seguimiento de un plan.

De estos cuatro valores, se desarrollaron doce principios para el manifiesto ágil y que son:

1. La satisfacción del cliente a través de la entrega rápida y continua de paquetes de software útiles y de valor.
2. Nuevos requisitos son bienvenidos incluso en la etapa final del desarrollo.
3. Entrega con frecuencia de software que funcione, preferentemente en semanas en vez de meses.
4. El software que funciona es la prueba fehaciente de que se puede medir el progreso del proyecto.
5. Desarrollo sostenible, capaz de mantener un ritmo constante.

6. Trabajo cercano de forma cotidiana entre las personas de negocio y desarrolladores.
7. La conversación cara a cara es la mejor forma de comunicación.
8. Los proyectos están construidos en un entorno de personas motivadas, a los cuales se les tiene que dar la confianza necesaria para que realicen la tarea.
9. Atención continúa a la excelencia técnica y al buen diseño.
10. Simplicidad.
11. Equipos que se auto organizan.
12. Adaptación regular a las circunstancias cambiantes.

El marco[xxiv] de desarrollo ágil más utilizado actualmente, es Scrum. Se presenta como contrapunto a PMBOK y PRINCE2, siendo utilizada tanto para desarrollo de software como para otro tipo de productos.

Por otra parte, también se disponen de metodologías específicas para el desarrollo de software que pretenden ser alternativas a estándares como ISO/IEC 15504, ISO/IEC 12207y CMMI. Por ejemplo:

- Dynamic Systems Development Method (DSDM): Metodología ágil más veterana y la que más se aproxima a los métodos tradicionales, su implantación incluso permitiría alcanzar un nivel 2 de madurez según CMMI.
- Extreme Programming (XP): La metodología ágil más radical y popular. XP se centra en el ciclo de vida del desarrollo de software.
- Agile Modeling: Metodología para el modelado y la generación de documentación, que se encuentra alineado con los principios del desarrollo ágil y que puede ser utilizado como substituto del UML estándar.
- Feature Driven Development (FCC): Metodología de desarrollo de software orientada a la generación de valor para el cliente.

Los roles propuestos por Scrum son: el Dueño de Producto, el Scrum Master y el Scrum Team.

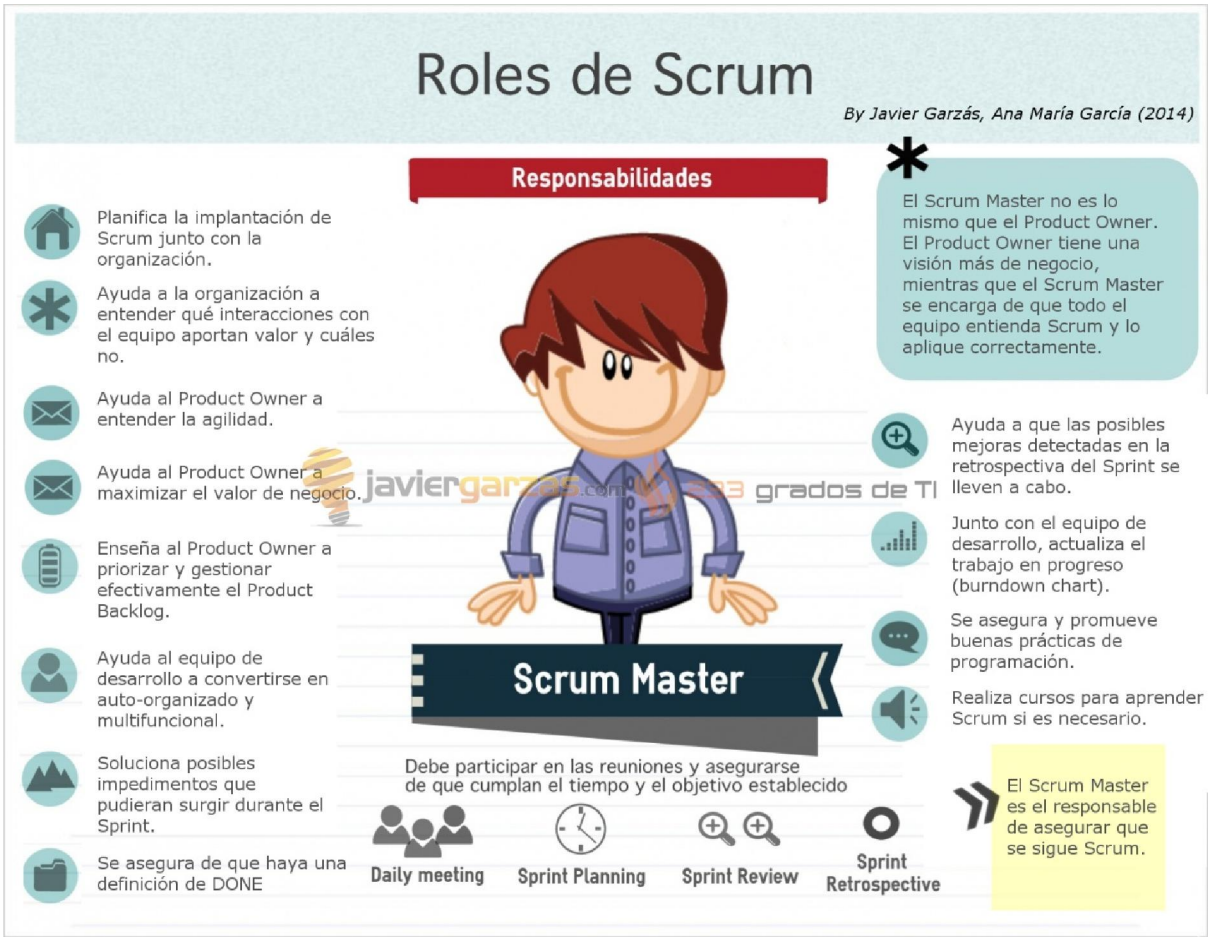


Ilustración 2-14 Roles SCRUM, Master[xxv]

El Dueño de Producto es la única persona autorizada para decidir sobre las funcionalidades y características funcionales tendrá el producto. Es quien representa al cliente, usuarios del software y todas aquellas partes interesadas en el producto.[xxvi]

El Scrum Master es la esencia de Scrum. No es un líder típico, sino que es un auténtico servidor neutral, que será el encargado de fomentar e instruir sobre los principios ágiles de Scrum.

El Scrum Team (o simplemente "equipo"), es el equipo de desarrolladores multidisciplinario, integrado por programadores, diseñadores, arquitectos, probadores y demás, que en forma auto-organizada, serán los encargados de desarrollar el producto.

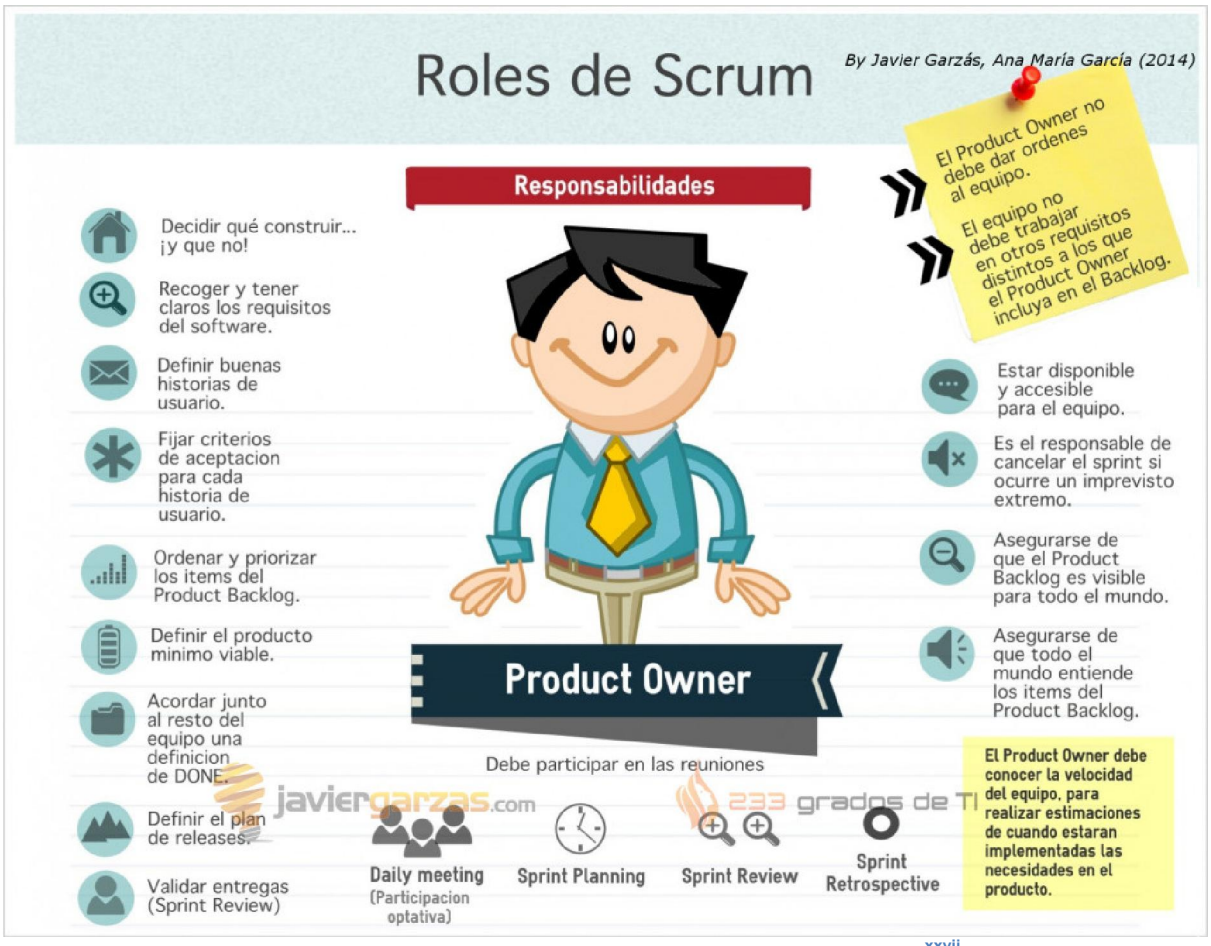


Ilustración 2-15 Responsabilidades del Rol: Product Owner[xxvii]

2.3 NEGOCIO

Los problemas de los sitios actuales son que ninguno tiene de todo, cada uno tiene sólo unas cosas, no aúnan toda la información posible, ni todas las posibilidades... Otro problema es que están llenos de publicidad, que al final despista y hace el uso del sitio bastante pesado. No tienen buscadores completos, algo que yo considero importante para analizar y entender bien las canciones y lo que quiere transmitir un grupo con sus letras. Algunas están en inglés, ninguna tiene traducciones correctas, y mucho menos resumen ni etiquetas identificativas, ni discusiones sobre las letras en redes sociales(o las que hay son pobres).

Algunas de la amplia lista de webs sobre letras de canciones son:

http://www.quedeletras.com/ Dispone las letras de las canciones, videoclips, discografías, biografías, pero con publicidad y poco organizado.

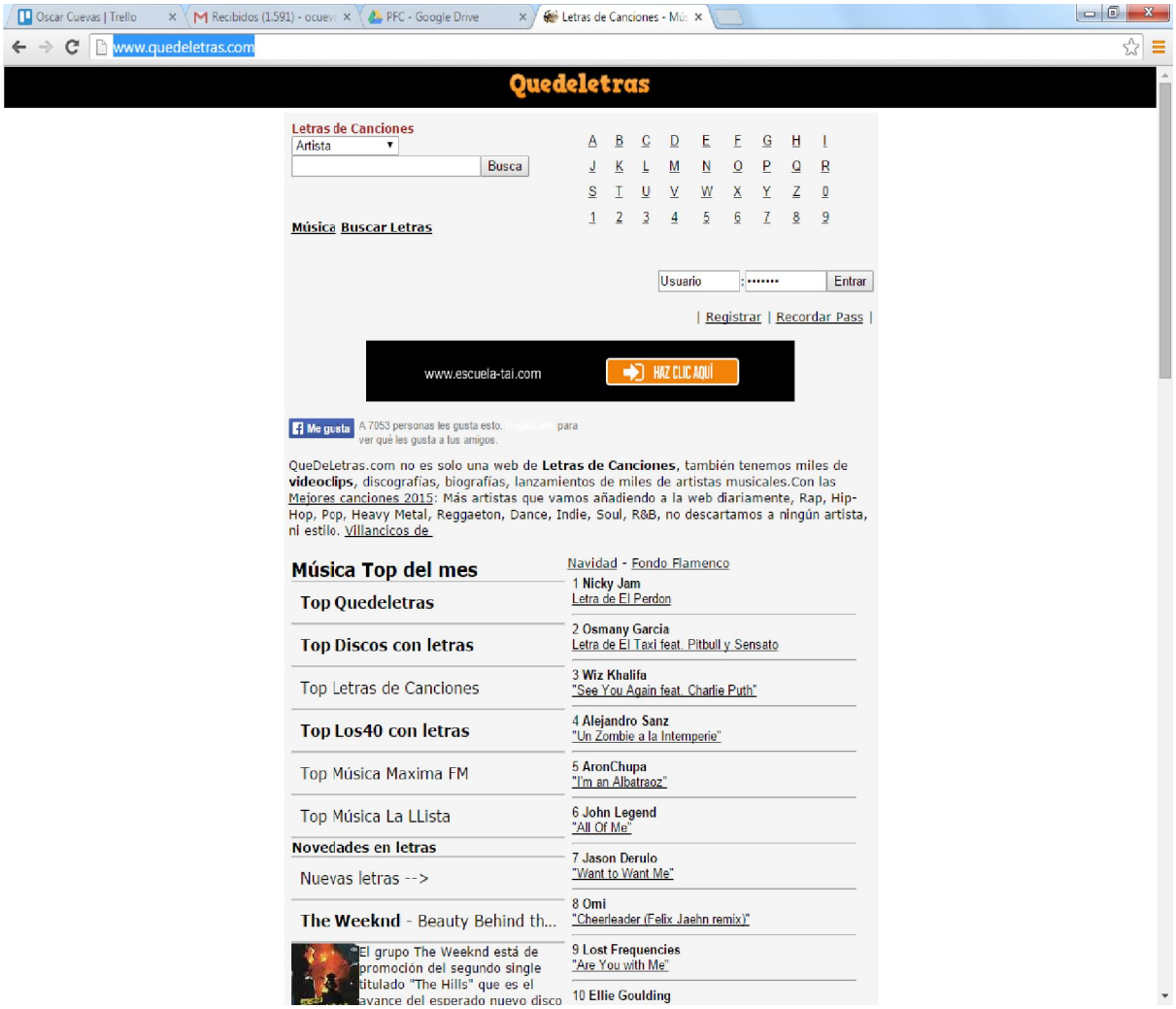


Ilustración 2-16 http://www.quedeletras.com/

http://www.dicelacancion.com/ Letras de canciones hasta vídeos musicales y noticias del mundo del espectáculo, así como curiosidades sobre los diferentes géneros musicales. Tiene publicidad y no tiene búsqueda avanzada.

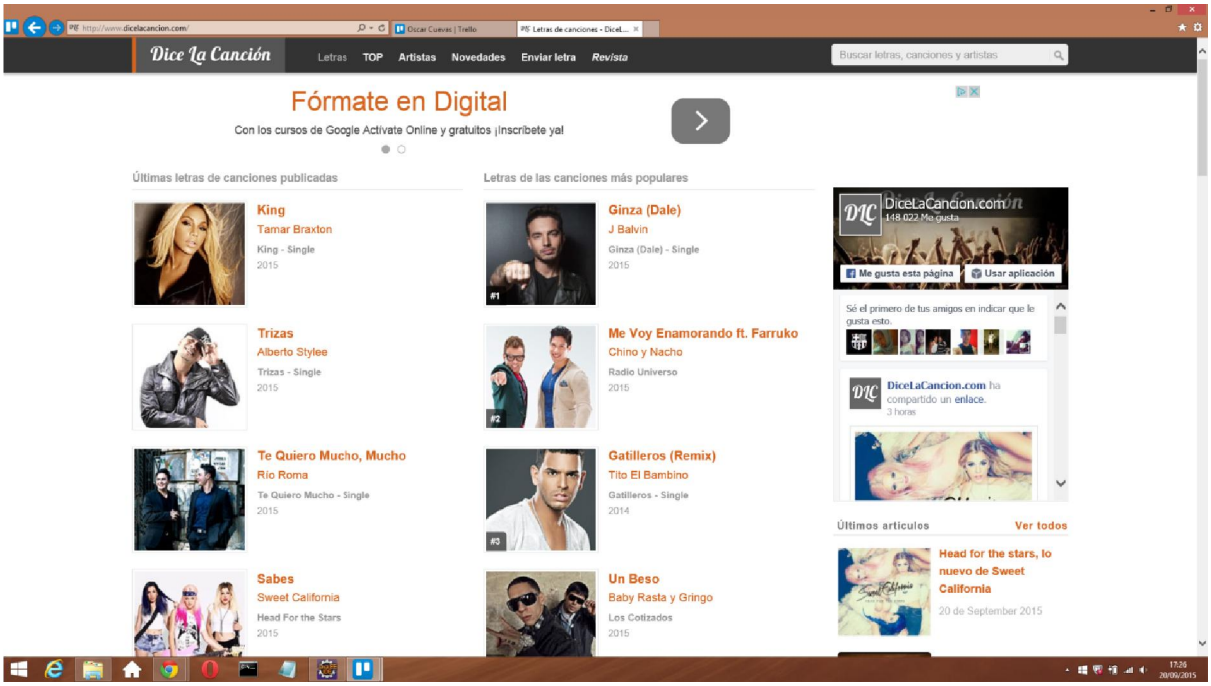


Ilustración 2-17 http://www.dicelacancion.com/

http://www.musica.com/ Tiene letras de canciones. Tiene mucha publicidad y sin buscador avanzado.

Ilustración 2-18 http://www.musica.com/

http://www.songstraducidas.com/ Letras y buscador, pero con publicidad y poco organizado.

Ilustración 2-19 http://www.songstraducidas.com/

Veamos ahora un listado del resultado de la búsqueda en Google de letras de canciones:

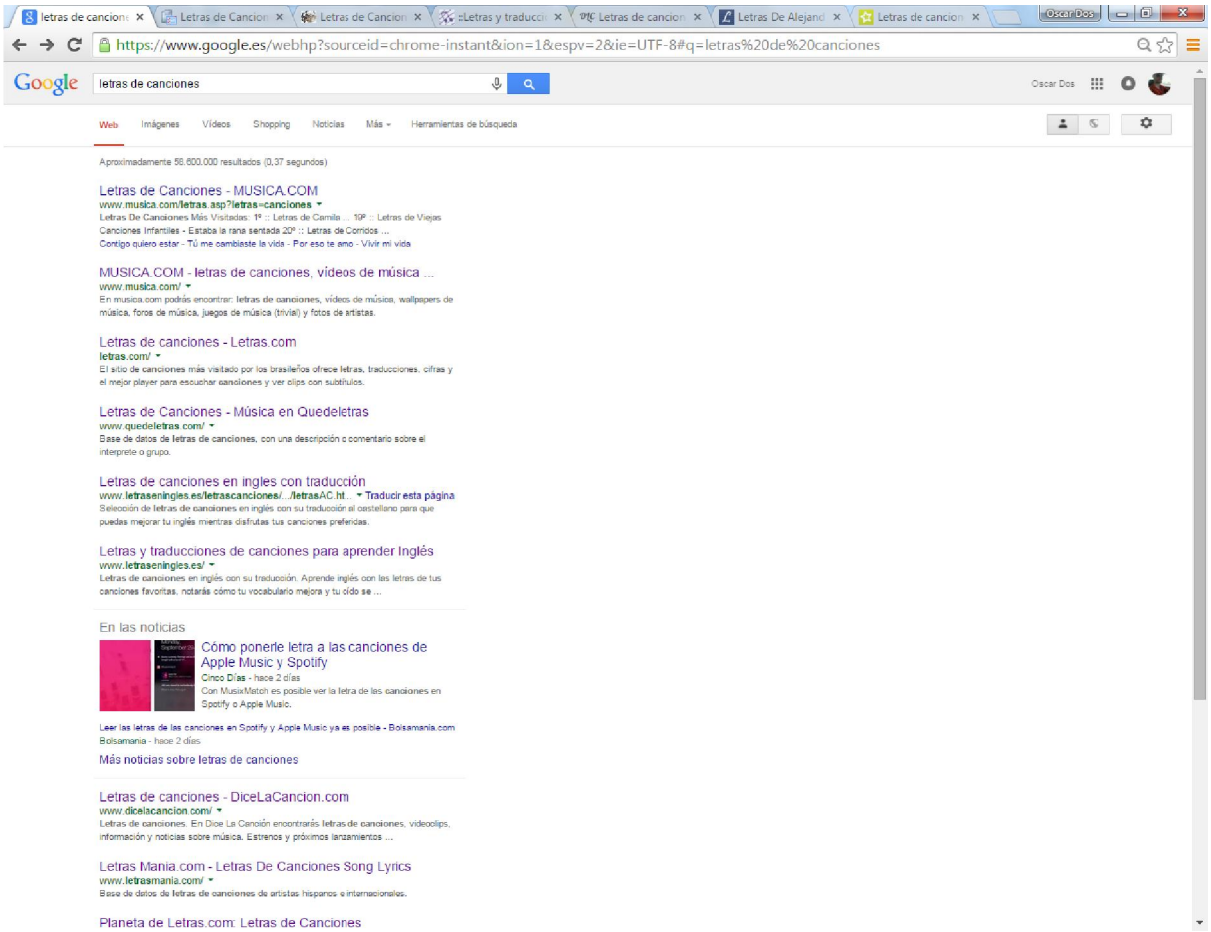


Ilustración 2-20 Búsqueda en Google: Letras de Canciones 1

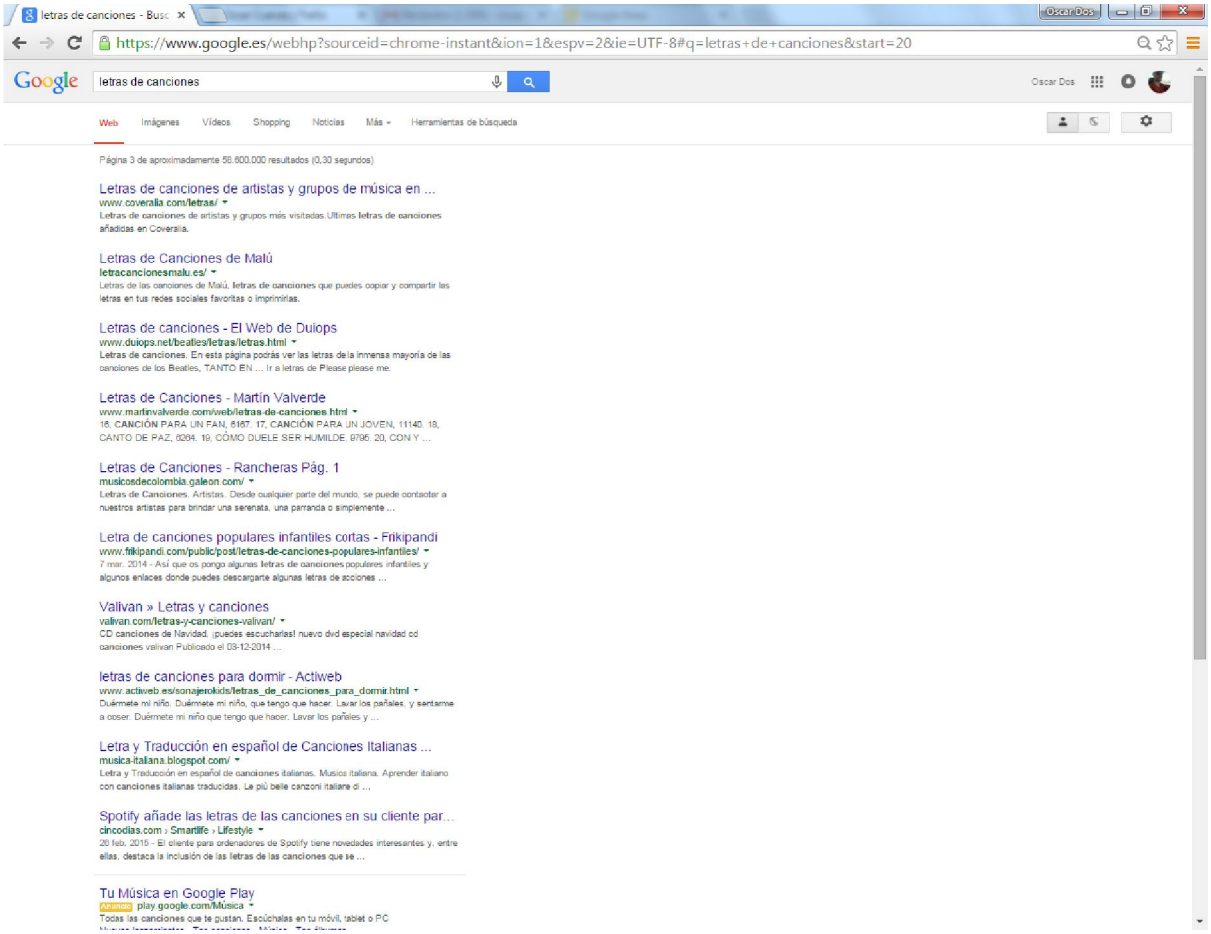


Ilustración 2-21 Búsqueda en Google: Letras de Canciones 2

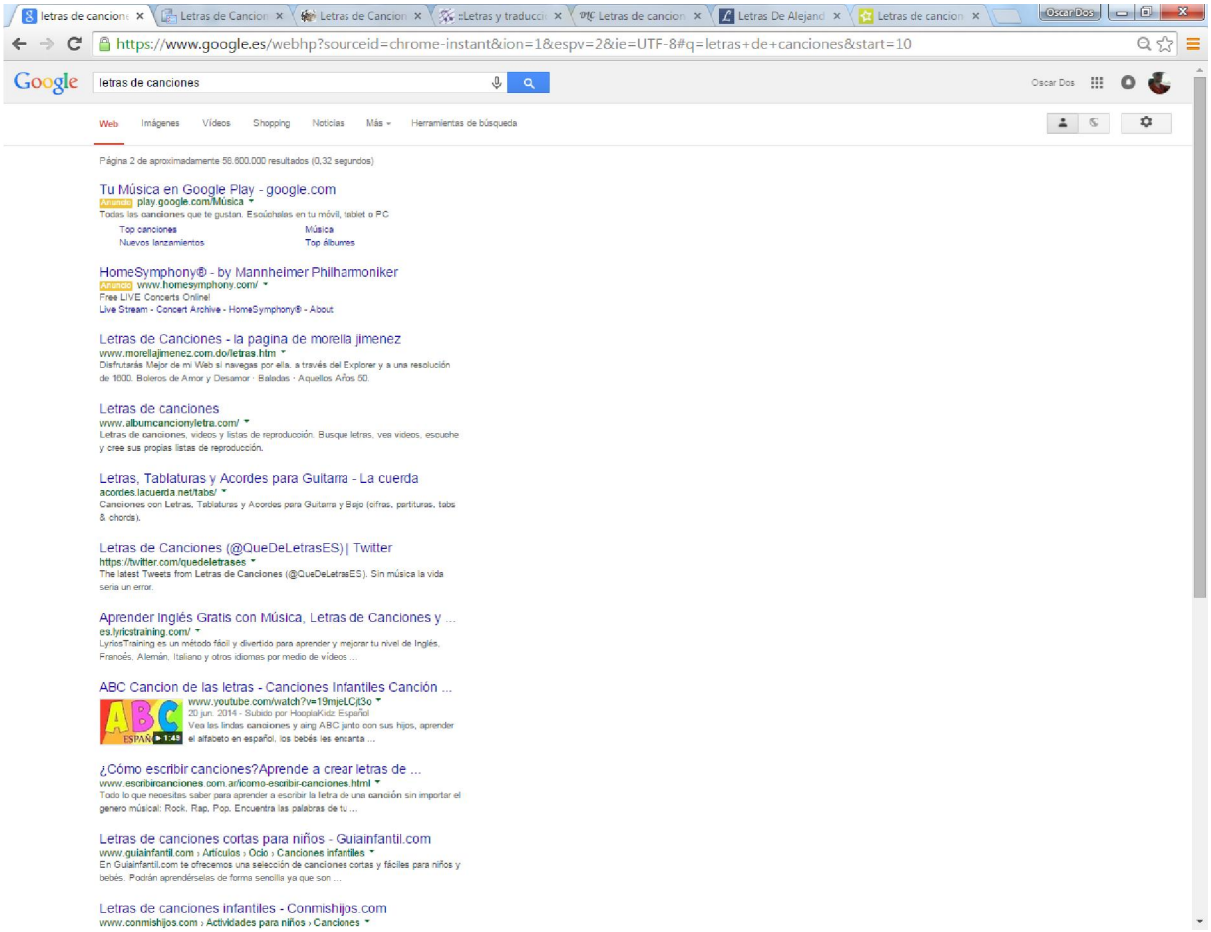


Ilustración 2-22 Búsqueda en Google: Letras de Canciones 3

Otras webs sobre la música y sus letras.

http://www.metrolyrics.com/
http://www.portaldelrock.com/
http://www.englishtown.com/es-mx/blog/canciones-ingles-significado/
http://www.sufridoresencasa.com/canciones-con-doble-sentido/
http://www.ideasnopalabras.com/
https://www.um.es/tonosdigital/znum2/estudios/ExtremoTonos2.htm
https://www.um.es/tonosdigital/znum2/estudios/ExtremoTonos2.htm
http://lacuerda.net/ Dispone de partituras y tablaturas de guitarra(TABS).
http://www.partiturasdeguitarraclasica.com/ Tiene TABS y partituras.
http://partiturasya.blogspot.com.es/
http://lyricsfly.com/
http://s3pu.blogspot.com.es/
http://www.ompersonal.com.ar/singinggrammar/
http://www.letrastraducida.com/
http://www.traduceletras.net/
http://www.traducidas.com.ar/
http://www.songstraducidas.com/

3. CAPÍTULO 3: ESTRATEGIA

En este capítulo se describe la arquitectura de implementación del sistema, el diseño de la aplicación entre el cliente y el servidor, la metodología seguida, las historias de usuario que el cliente desea para su aplicación y el sistema de pruebas para verificar el correcto funcionamiento de todo el conjunto.

3.1 INTRODUCCIÓN

3.1.1 ¿QUÉ ES UNA WEB APP?

Web App es la manera de llamar habitualmente a una aplicación web, en referencia a su nombre en inglés: *Web Aplication*. Son aplicaciones desarrolladas para funcionar desde un navegador Web, algunas de las cuales pueden trabajar del lado del cliente o bien conectarse e interactuar con tecnologías del lado servidor, para intercambiar datos o realizar otras operaciones.

A partir de mediados de la década de los noventa, con la aparición de JavaScript y ciertas tecnologías que comienzan a integrarse en los navegadores, surgen los primeros ejemplos de cómo se puede crear interacción mediante *scripts* desde el lado del cliente.

Casi una década después, AJAX combinaría todas las técnicas existentes para permitir la creación de aplicaciones Web de la talla de Gmail y Google Maps, entre otras. También abriría las puertas para funciones indispensables para redes sociales como Facebook o Twitter.

Lo último es el HTML5, que es un conjunto de tecnologías que nos permiten crear aplicaciones que funcionan de manera nativa, tanto en navegadores de escritorio como en móviles, sin necesidad de instalar *plugins* o incluir agregados de terceros. Esto abre una nueva era para las aplicaciones Web, con características que van desde las posibilidades multimedia y el 3D hasta el almacenamiento local del lado cliente y la posibilidad de trabajar de manera *offline*.

3.1.2 APLICACIONES PARA MÓVILES: NATIVAS, WEB O HÍBRIDAS

A la hora de elegir cómo crear una aplicación para móvil, hay que tener en cuenta las características y alcances que tendrá el desarrollo que se esté planificando, así como las necesidades, recursos disponibles, previsión de actualizaciones futuras, etc.

Una aplicación **nativa** cuenta con las ventajas de permitir una adaptación pensada para cada plataforma, dándonos la posibilidad de crear alternativas distintas, si fuera necesario. El acceso al hardware está disponible y nos permite una manera eficiente de interactuar con los recursos del

dispositivo. Para desarrollarla hay que conocer el lenguaje de programación adecuado para cada plataforma: al crear soluciones para diferentes sistemas, deberemos dominar varios lenguajes.

Si desarrollamos una aplicación **Web** hospedada en Internet contamos con la ventaja de tener mayor control y facilidades para la actualización sin que el cliente deba realizar ninguna acción. Se cambia todo en el servidor, sin necesidad de modificar nada en el equipo del usuario.

Una novedad que introduce HTML5 es la posibilidad de que funcionen *offline* y utilicen almacenamiento local.

También existen las aplicaciones denominadas **híbridas**. Éstas se escriben con los lenguajes empleados en el mundo Web (HTML, CSS y JavaScript) y luego se empaquetan como nativas para plataformas móviles. Se pueden publicar y distribuir desde las tiendas *online* y el usuario las podrá descargar e instalar.

La idea de este modelo de aplicación se apoya en el concepto de escribir el código base de una vez y después poder emplearlo en diversas plataformas, con modificaciones mínimas que ahorran muchas horas de desarrollo y pueden permitir reducir costes en los proyectos. Es una buena opción para salvar el dilema de la fragmentación de sistemas operativos y dispositivos existentes en el mercado.

Igual que encontramos mucha variedad en los modelos de dispositivos, también hay mucha variedad en los sistemas operativos para móviles.

El sistema operativo en los móviles define varias características importantes para un desarrollo, ya que pueden cambiar las herramientas y el lenguaje con el cual se generan las aplicaciones nativas. Si pensamos en el desarrollo Web para móviles, podremos observar que cada sistema operativo cuenta con un navegador que nos llega de fábrica como predeterminado, y que nos encontramos con la posibilidad de optar por instalar alternativas de otros desarrolladores, si lo deseamos.

3.1.3 SISTEMAS OPERATIVOS MÓVILES

Los principales sistemas operativos para *smartphones* y *tablets* son:

- **Android:** su desarrollo está en manos de Open Handset Alliance, alianza con más de ochenta empresas, con Google como actor principal en este proyecto. Android es el sistema operativo más utilizado en la actualidad por *smartphones* y *tablets,* con más de la mitad del mercado. Cuenta con adaptaciones para dispositivos de diversos fabricantes. Un dato para pensar si vamos a desarrollar para Android es las versiones que aún están vigentes y el API Level de cada una de ellas.

- **iOS:** es el sistema que Apple creó para sus dispositivos móviles, originalmente par iPhone, y luego lo extendió para iPod Touch y también para iPad. Es el segundo sistema operativo

móvil más utilizado, con algo más de un cuarto del mercado. Es importante destacar que iOS es un sistema que sólo funciona en dispositivos Apple y no se encuentra disponible para otros fabricantes.

- **Symbian:** es un sistema con una larga tradición en el mundo de los móviles. NOKIA, Sony Ericsson y Motorola desarrollaron por años muchos dispositivos exitosos en el mercado que se han apoyado en la confiabilidad de Symbian. Sin embargo, algunas decisiones estratégicas, como la de Nokia eligiendo Windows Phone para sus nuevos dispositivos están marcando el principio del fin para Slymbian.

- **Windows Phone:** es el sistema que actualmente desarrolla Microsoft para dispositivos móviles. Ha sido adaptado por dispositivos de diversas empresas de móviles en todo el mundo, entre ellas: Hacer, Dell, HTC, Nokia, Samsung y LG. Previo al lanzamiento de Windows Phone, Microsoft desarrolló otros sistemas tales como Windows CE y Windows Mobile. Estas versiones antiguas ya no cuentan con muchas unidades en el mercado activo y sus aplicaciones no son compatibles con las del nuevo sistema operativo.

- **BlackBerry OS:** es la denominación del sistema operativo instalado en la línea de teléfonos BlackBerry. Su nacimiento data de fines de los años noventa y sus diferentes versiones acompañaron la evolución de los *smartphones* de RIM, empresa que desde el año 2013 pasó a llamarse simplemente BlackBerry. La llegada de la *tablet* PlayBook, en el año 2011, también marcó el arribo de un nuevo sistema operativo: BlackBerry Tablet OS.

- **WebOS:** es un sistema creado por Palm en el año 2009. Posteriormente pasa a manos de Hewlett-Packard bajo el nombre de HP webOS. En 2011 se anunció que no se continuaría con la fabricación de dispositivos que incluyan este sistema operativo, sin embargo, se mantiene el soporte y se ha modificado su modalidad de licencia para que sea software libre.

- **Firefox OS:** es el nombre con el que se ha lanzado el sistema operativo preparado por Mozilla Corporation. Es desarrollado bajo el modelo código abierto, se basa en núcleo Linux, apoyándose en las virtudes del motor Gecko (el mismo que emplea Firefox). Se destaca por ser una opción liviana para dispositivos de hardware limitado. Su fortaleza de vasa en los estándares Web y su capacidad de correr aplicaciones creadas en HTML5.

En el mercado, podremos encontrar otros sistemas operativos para móviles, como el caso de MeGoo (sucesor de Maemo y Moblin) y Bada. Vale la pena destacar también que existen distribuciones de Linux adaptadas como sistemas operativos para móviles. Esta última opción se puede ver con mayor presencia en mercados asiáticos, como los casos de China y Japón.

3.2 ARQUITECTURA

3.2.1 INTRODUCCIÓN

3.2.1.1 EL MOMENTO DE HTML5

La última especificación de HTML4 es del año 1999, a partir de ahí se decidió avanzar hacia la quinta versión, que empezó a mostrar sus avances en los navegadores más populares entre los años 2009 y 2010.

HTML5 reconoce compatibilidad con HTML4, quitando del estándar sólo aquellas etiquetas que ya no resultan útiles (obsoletas), modificando los elementos que necesitan actualizarse y agregando todo un nuevo mundo de posibilidades. Llega una nueva etapa en lo que respecta a compatibilidad multiplataforma y con las diferentes versiones de los navegadores.

3.2.1.2 EL NUEVO ROL DE JAVASCRIPT

Al principio JavaScript tenía algunos efectos y características limitadas, pero con la evolución de la Web y la llegada de AJAX, JavaScript tuvo un renacer que redefinió su rol en el mundo de Internet.

El HTML5 le da una nueva vida y papel protagonista a este lenguaje. Un cambio importante en el enfoque de JavaScript está dado en que ya no debe considerase sólo un lenguaje para trabajar del lado del cliente, sino que también ofrece opciones para tecnologías del lado del servidor.

La realización de este proyecto engloba el diseño y construcción de un sistema cliente-servidor. El sistema cliente realmente es el navegador elegido por el usuario con cualquier dispositivo (hecho ya el análisis entre aplicaciones de escritorio y aplicaciones web). Y para el sistema servidor no está definida la arquitectura a usar, de tal manera que se analizarán las diferentes arquitecturas disponibles en el mercado para escoger la más adecuada en base a unos criterios.

3.2.1.3 JAVA SERVER PAGES

La tecnología de las Java Server Pages (JSP) aparece para aliviar el engorro que supone generar páginas web mediante *servlets*. Además de esta forma es posible dividir el trabajo de construir una aplicación web entre programadores (servlets) y diseñadores gráficos (JSPs).

Una página JSP no es más que una página HTML que contiene ciertas etiquetas especiales (acciones) o fragmentos de código (scriptlets) para incluir contenido dinámico en dichas páginas. El código HTML se añade tal cual a la respuesta, mientras que los elementos dinámicos se

evalúan con cada petición. Cuando un cliente pide esa página al servidor de aplicaciones, se traduce a un fichero fuente de Java que implementa un Servlet, se compila y se ejecuta para devolver el resultado de la petición.

Una vez compilado el servlet, las peticiones posteriores se reducen a ejecutar dicho servlet, en lugar de realizar el proceso una y otra vez. Esta es la razón por la que la primera petición tarda mucho más en responderse que las siguientes.

Dado que una JSP no es más que un servlet, hay que recordar que todo el código se ejecuta en el servidor, las respuestas son puramente páginas HTML.

3.2.1.4 SCRIPLETS

Los Scriptlets no son más que fragmentos de código delimitados entre <% y %>, que se ejecutan para resolver la petición de un usuario.

Hay un tipo especial de Scriplet delimitado entre <%= y %>. La expresión contenida se calcula y el resultado se añade a la respuesta. Es decir, <%= user.getName() %> es exactamente igual a: <% out.println(user.getName()); %>.

Twitter Bootstrap es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Ha sido desarrollado por Twitter, y la mayor ventaja que aporta es que se pueden crear interfaces que se adapten a los distintos navegadores (diseño responsivo), de forma sencilla y rápida.

3.2.2 COMPARACIÓN DE ARQUITECTURAS

Pasemos ahora a ver las opciones de alojamiento que se han analizado, y las distintas combinaciones de arquitecturas disponibles: One.com o Google App Engine.

A la hora de construir el sistema servidor, hay que tener en cuenta de que se tiene acceso a un alojamiento web en los servidores de One.com por lo que habrá que analizar los servicios que ofrece esa compañía. Principales servicios de One.com:

15 GB de almacenamiento: cantidad de espacio para almacenar los datos en el servidor.
Tráfico de Internet ilimitado: tráfico tanto de descarga de datos, como de subida de datos al servidor ilimitado.
PHP: lenguaje de programación del lado del servidor originariamente creado para desarrollo web de contenido dinámico. Se le considera uno de los lenguajes más potentes, flexibles y de alto rendimiento conocido a día de hoy.
MySQL: sistema gestor de bases de datos relacionales para almacenar los datos de los usuarios.

SSL: protocolo para proporcionar comunicaciones seguras a través de la red.

Una vez que se han analizado los servicios que ofrece la compañía One.com, se quiere realizar un sistema servidor cuya arquitectura implique un desembolso económico a coste cero y posea un alto rendimiento. Es por este motivo que se pretende analizar dos arquitecturas para elegir la que más se adapte a las necesidades económicas y de rendimiento.

3.2.2.1 DEFINICIÓN DE LA ARQUITECTURA A

La Arquitectura A estaría alojada en los servidores web de One.com y tendría un coste de despliegue nulo. Está compuesta por las tecnologías que se tienen acceso en los servidores webs de la compañía (PHP y MySQL) en combinación con soluciones ágiles para el uso en dispositivos móviles (JSON y REST) y un framework liviano para interactuar con la tecnología REST.

PHP: lenguaje soportado por el servidor de alojamiento. En combinación con una base de datos MySQL se puede construir un sistema rápido y eficaz.

MySQL: al disponer de alojamiento gratuito en el servidor web y soportar este tipo de base de datos, se ha elegido como base de datos de esta posible arquitectura.

JSON: se ha elegido por la simplicidad en el intercambio de información que se produce entre cliente y servidor.

REST: arquitectura para la comunicación entre cliente y servidor. Dado que REST trabaja con el protocolo HTTP, la simplicidad que posee en comparación con otros métodos de comunicación como CORBA, SOAP o WSDL hace que su uso sea muy utilizado en los sistemas de información de hoy en día.

SLIM Framework: marco de desarrollo probado y testeado que facilita la creación de APIs.

REST: su facilidad de uso, limpieza, ligereza, soporte de todos los métodos HTTP (GET, POST, PUT, DELETE) y la capa intermedia para filtrar las peticiones, le hacen un marco imprescindible a la hora de crear un API REST.

3.2.2.2 DEFINICIÓN DE LA ARQUITECTURA B

Esta arquitectura estaría implementada con las herramientas que proporciona Google para la construcción de APIs. Se almacenaría en los servidores de Google en lo que se llama “Google Cloud Platform” haciendo uso de su tecnología App Engine. Google Cloud Endpoint: solución de Google para la creación de APIs a través de su servicio App Engine facilita la tarea para la integración con aplicaciones móviles Android e iOS.

JAVA: lenguaje de programación concurrente basado en objetos soportado por los servicios de Google Cloud Platform. Elegido para la posible Arquitectura B frente a Python por el conocimiento del lenguaje.

GOOGLE CLOUD DATASTORE: base de datos no relacional (noSQL) donde se almacenan los datos guardados en Google Cloud Platform.

JDO: capa de persistencia de Java para salvar los datos en los servidores en la nube de Google.

REST: arquitectura de comunicación usada para la comunicación entre el cliente y servidor.

Arquitectura A. PHP + MySQL + JSON + REST + Slim Framework

Arquitectura B. Google App Engine + Java + JDO + REST

Google App Engine es un servicio que permite ejecutar las aplicaciones web utilizando la infraestructura de Google. Los lenguajes soportados por Google App Engine son Python, Java, Go y PHP. Características principales:

1. ALMACÉN DE DATOS

App Engine proporciona un potente servicio de almacenamiento de datos distribuido que además incluye un motor de búsqueda y de transacciones. El almacén de datos es de consistencia fuerte y utiliza el control de concurrencia optimista. Por encima de todo se garantiza la integridad de los datos.

2. CUENTAS DE GOOGLE

Evidentemente App Engine admite la integración de las aplicaciones con Google Accounts para la autenticación de usuarios. Acceder a tus aplicaciones con las cuentas de Google Accounts es un avance, ya que permite a los usuarios acceder a las aplicaciones de una forma más rápida.

También mirando el desarrollo, evitar el diseño e implementación de un sistema de cuentas de usuario, lo que es un ahorro importante en tiempo (y en dinero).

La autenticación en Google se realiza utilizando Oauth 2.0[xxviii].

OAuth2 es un protocolo de autorización que permite a terceros (clientes) acceder a contenidos propiedad de un usuario (alojados en aplicaciones de confianza, servidor de recursos) sin que éstos tengan que manejar ni conocer las credenciales del usuario. Es decir, aplicaciones de terceros pueden acceder a contenidos propiedad del usuario, pero estas aplicaciones no conocen las credenciales de autenticación .

En un escenario OAuth2 hay tres partes claramente identificadas:

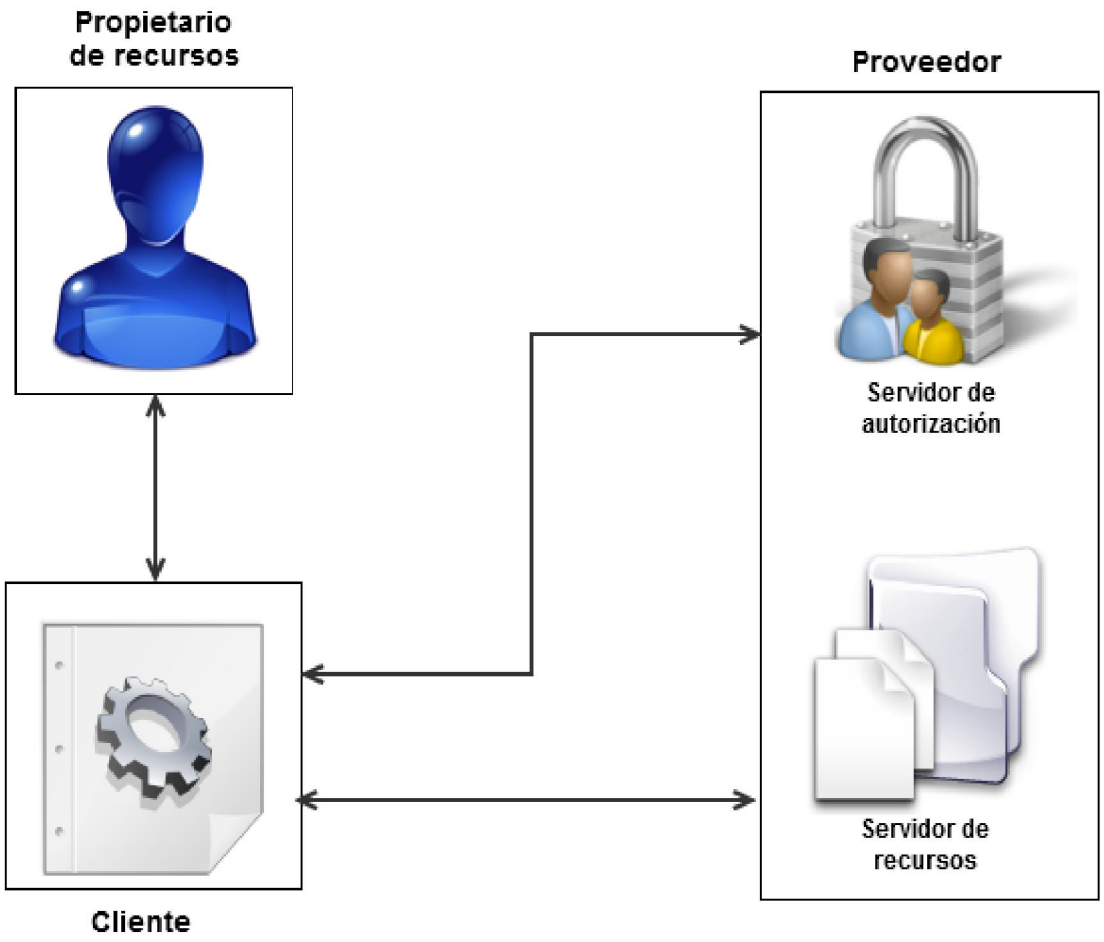


Ilustración 3-1 Elementos implicados en Oauth2[xxix]

- **Propietario de recursos**. Es una entidad capaz de dar acceso a recursos protegidos. Cuando es una persona nos referiremos a él como usuario final.
- **Cliente**. Es la aplicación que hace peticiones a recursos protegidos en nombre de un propietario de recursos con la autorización del mismo.
- **Proveedor.** Entidad que provee el servicio de autenticación.
- **Servidor de recursos**. Es la entidad que tiene los recursos protegidos. Es capaz de aceptar y responder peticiones usando un access token que debe venir en el cuerpo de la petición.
- **Servidor de autorización**. En muchos casos el servidor de autenticación es el mismo que el Servidor de Recursos. En el caso de que se separen, el servidor de autenticación es el responsable de generar tokens de acceso y validar usuarios y credenciales.

Con este escenario surge la necesidad de implementar un protocolo de autorización, en el que el usuario final pueda autorizar a aplicaciones de terceros (consumidores) a acceder a sus datos en la aplicación proveedora sin necesidad de darle sus credenciales.

3. SERVICIOS

Otro de los puntos a destacar son algunas características que App Engine proporciona y que nos facilitan el trabajo al administrar nuestra aplicación:

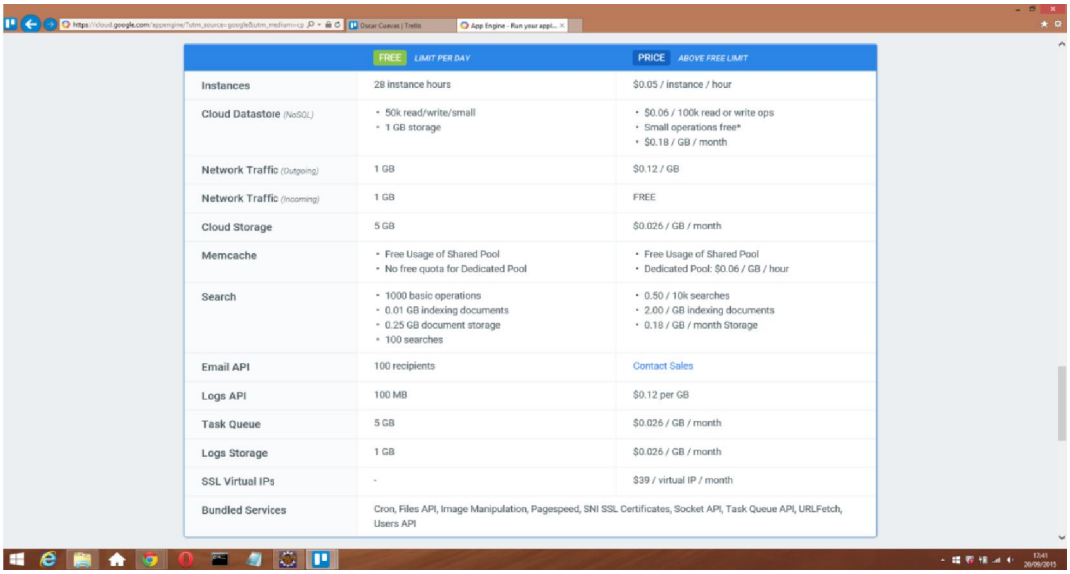
- **Extracción de URL:** Recupera recursos web mediante la misma infraestructura de alta velocidad de Google.
- **Correo:** La App puede enviar correos electrónicos a los usuarios por medio de las herramientas de Google.
- **Memcache**: App Engine proporciona una memoria caché de valores-claves de alto rendimiento accesible desde varias instancias de tu aplicación. Resulta útil para datos que no necesitan las funciones de persistencia y transacciones del almacén de datos. Velocidad.
- **Manipular imágenes:** Recortar, girar o ajustar imágenes desde tu aplicación de una forma sencilla.

4. JAVA

Google App Engine admite las herramientas de desarrollo web Java y de estándares del API conocidos. El SDK Java de App Engine permite desarrollar aplicaciones que utilicen Java 7.

5. CUOTAS

Además de sencillo, App Engine es gratis. Crear una cuenta, publicar tu App y que otros usuarios puedan utilizarla al momento, no tiene ningún coste, es gratis. El ‘ paquete gratuito’ dispone de 500 MB de espacio para tu App y admite hasta 5 millones de visitas mensuales. Si llega el momento en el que necesitas facturar, puedes habilitarlo y establecer un presupuesto diario máximo y asignarlo para cada recurso según las necesidades.



	FREE LIMIT PER DAY	PRICE ABOVE FREE LIMIT
Instances	28 instance hours	$0.05 / instance / hour
Cloud Datastore (NoSQL)	• 50k read/write/small • 1 GB storage	• $0.06 / 100k read or write ops • Small operations free* • $0.18 / GB / month
Network Traffic (Outgoing)	1 GB	$0.12 / GB
Network Traffic (Incoming)	1 GB	FREE
Cloud Storage	5 GB	$0.026 / GB / month
Memcache	• Free Usage of Shared Pool • No free quota for Dedicated Pool	• Free Usage of Shared Pool • Dedicated Pool: $0.06 / GB / hour
Search	• 1000 basic operations • 0.01 GB indexing documents • 0.25 GB document storage • 100 searches	• 0.50 / 10k searches • 2.00 / GB indexing documents • 0.18 / GB / month Storage
Email API	100 recipients	Contact Sales
Logs API	100 MB	$0.12 per GB
Task Queue	5 GB	$0.026 / GB / month
Logs Storage	1 GB	$0.026 / GB / month
SSL Virtual IPs	-	$39 / virtual IP / month
Bundled Services	Cron, Files API, Image Manipulation, Pagespeed, SNI SSL Certificates, Socket API, Task Queue API, URLFetch, Users API	

Ilustración 3-2 Cuotas de una cuenta gratis GAE y precios si se superan

6. ESCALABLE

Una de las principales cosas a tener en cuenta al hacer una App es la escalabilidad. El que sea más fácilmente escalable es un punto a tener muy en cuenta. Google App Engine destaca por ello, al igual que por la estabilidad y por la seguridad de 'nuestras' Apps.

Desde el punto de vista de los clientes de servicios, y no de los usuarios desarrolladores, el uso de GAE no requiere la instalación de ningún programa especifico (se trata de aplicaciones web accesibles desde un navegador cualquiera) y es totalmente gratuito y transparente. Por su parte, el usuario desarrollador es el encargado de habilitar una cantidad determinada de recursos disponibles para sus aplicaciones. Existe una cuota de recursos gratuita, pero si se desea permitir el consumo de recursos adicionales, es necesario activar una cuenta de facturación y en ese caso el usuario desarrollador pagará por todos aquellos recursos utilizados por los clientes de servicios, que excedan los límites de las cuentas gratuitas.

Los recursos que se encuentran disponibles van desde capacidad de almacenamiento extra, mayor ancho de banda o tiempo de CPU superior. Por otra parte, a día de hoy, por cada cuenta de usuario Google no es posible alojar más de 20 aplicaciones diferentes. GAE fue lanzado en Abril de 2008 en versión beta con soporte en exclusiva para Python y, más tarde, incorporó también el lenguaje de programación Java. Sin embargo, Google impone una serie de restricciones en el uso de dicho lenguaje: los hilos, las conexiones de red directas y el código nativo, no están habilitados. Actualmente GAE se encuentra en constante desarrollo, con un número creciente de funcionalidades ofrecidas. Además, Google planea dar soporte a más lenguajes en el futuro ya que concibe la plataforma como independiente del lenguaje.

Java[xxx] es un lenguaje de programación orientado a objetos ampliamente extendido, desarrollado por Sun Microsystems[xxxi] a principios de los años 90. En la actualidad, Java ha pasado a ser propiedad de Oracle Corporation[xxxii]. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina la gestión a bajo nivel de los elementos, que suele inducir a muchos errores, como la manipulación directa de punteros o memoria. GAE permite lanzar aplicaciones en dos entornos de ejecución diferentes: el entorno Java y el entorno Python. Cada uno de ellos proporciona una serie de protocolos estándar y tecnologías propias para el desarrollo de aplicaciones web, como por ejemplo las Java Server Pages (JSP) en Java. A pesar de haber aparecido con anterioridad el entorno Python, la popularidad del lenguaje Java ha provocado que sus versiones del "binding" o adaptador para GAE hayan alcanzado un alto grado de convergencia, y ofrezcan en estos momentos prácticamente las mismas funcionalidades. Para el desarrollo de las aplicaciones de extracción en el entorno "cloud" se ha optado por utilizar el entorno de ejecución Java, puesto que ya se disponía de unos conocimientos previos muy avanzados de dicho lenguaje.

Eclipse[xxxiii] es un entorno de desarrollo integrado (IDE de sus siglas en inglés) de código abierto multiplataforma para desarrolladores. Fue concebido originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Actualmente es propiedad de la Eclipse Foundation, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Las principales características de Eclipse son: dispone de un editor de texto con resaltado de sintaxis, la compilación es en tiempo real, posee pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes para creación de proyectos, clases, tests, etc., y refactorización. Asimismo, a través de infinidad "plugins" o complementos libremente disponibles es posible añadir nuevas funcionalidades como el control de versiones con sub-versión. El desarrollo del código del sistema se ha realizado mediante la herramienta Eclipse, por el gran número de ventajas anteriormente descritas. Igualmente, para la codificación de los módulos pertenecientes al entorno “cloud” se ha hecho uso del “plugin” para GAE.

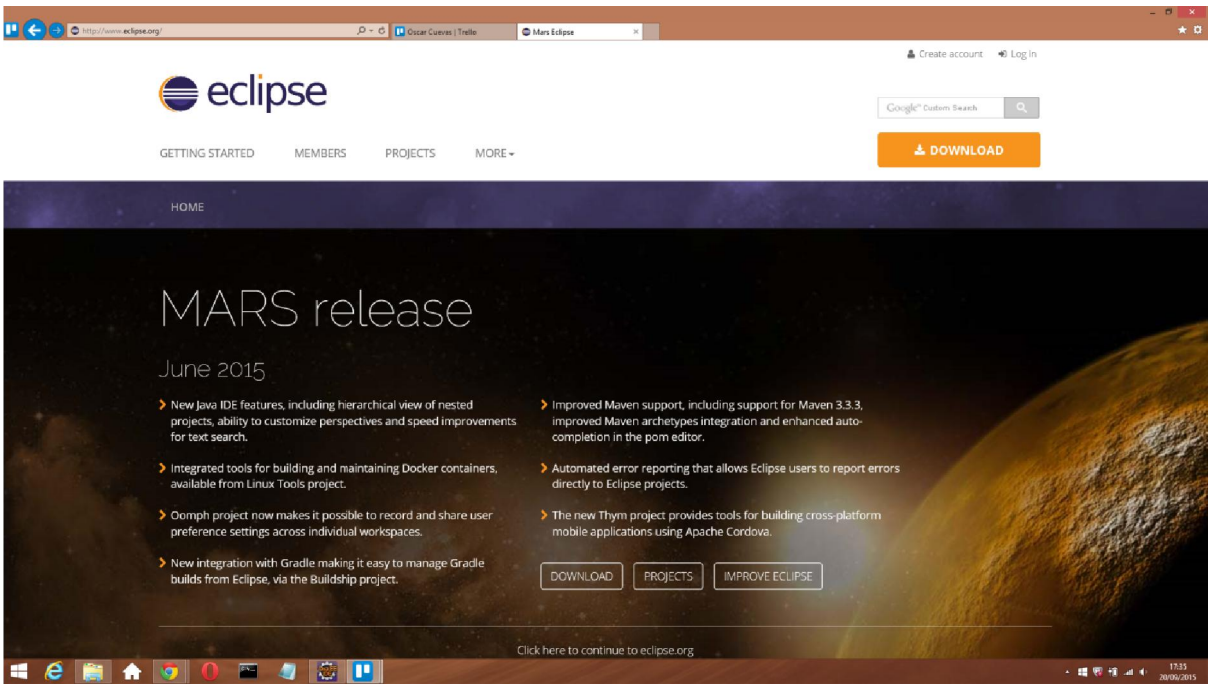


Ilustración 3-3 http://www.eclipse.org/

GAE Plugin para Eclipse[xxxiv]. El “plugin” de GAE[xxxv] para Eclipse permite desarrollar aplicaciones de App Engine, de forma ágil y eficaz. Incorpora unos botones especiales en la barra de herramientas, para facilitar la creación, compilación y despliegue de las aplicaciones.

El complemento está desarrollado por Google y es objeto de frecuentes mejoras y actualizaciones. La versión utilizada en la codificación del proyecto ha sido la Google Plugin for Eclipse 3.4.

Almacén de Datos de GAE. El almacén de datos es empleado por GAE para dotar a las aplicaciones de un entorno de almacenamiento escalable. Se ha diseñado para optimizar la lectura y la realización de consultas, y puede ejecutar varias operaciones en una única transacción recuperando la transacción entera si falla cualquiera de las operaciones. Esto resulta especialmente útil en el caso de las aplicaciones web distribuidas, en las que es posible que varios usuarios accedan a los mismos objetos de datos de forma concurrente.

A diferencia de las bases de datos tradicionales, el almacén de datos utiliza una arquitectura distribuida para gestionar el cambio de tamaño de los conjuntos de datos muy grandes. Una aplicación App Engine puede optimizar la forma en que se distribuyen los datos mediante la descripción de relaciones entre objetos de datos y la definición de índices para las consultas. El almacén de datos de App Engine es totalmente consistente, pero ello no implica que se trate de una base de datos relacional. Aunque su interfaz tiene muchas de las funciones de las bases de datos tradicionales, las características únicas del almacén de datos suponen una forma diferente de diseñar y gestionar los datos. Por ejemplo, las entidades del almacén de datos no tienen un esquema único: dos entidades del mismo tipo no están obligadas a contener las mismas propiedades o utilizar los mismos tipos de valores para las mismas propiedades. En consecuencia, la propia aplicación es la única responsable de garantizar que las entidades cumplan un determinado esquema cuando sea necesario.

Java Data Objects (JDO)[xxxvi] Google App Engine, a través del entorno de desarrollo SDK Java, incluye implementaciones de los Objetos de Datos Java (JDO) y de las Interfaces del API de persistencia de Java (JPA) para crear y persistir datos. En el desarrollo del proyecto se ha optado por utilizar Java Data Objects para la persistencia de los datos. JDO ofrece varias ventajas con respecto a JPA, tal y como puede apreciarse en la siguiente tabla comparativa, y dispone de una mayor cantidad de documentación para su uso dentro de GAE. Ésta última, junto con el mayor grado de desarrollo en GAE, han sido las razones fundamentales por las que finalmente se seleccionó JDO en detrimento de JPA.

	Java Data Objects (JDO)	***Java Persistence API (JPA)***
Requisitos *SDK Java*	1.3 ó superior	1.5 ó superior
RDBM (Relational Data Base Modeling)	Soportado	Soportado
ORM (Object Role Modeling)	Soportado	No soportado
Catálogo de tipos de datos soportados	Grande	Mediano
DataNucleus	Soportado	No soportado
Grado de desarrollo en *GAE*	Alto	Medio-Bajo

Ilustración 3-4 Comparación entre JDO y JPA

VENTAJAS Y DESVENTAJAS DE CLOUD COMPUTING

A primera vista una tecnología como Cloud Computing modifica los modelos de negocio existentes aportando numerosas ventajas tanto a los proveedores de los servicios de la nube como a los usuarios de esos servicios. Por ejemplo, a los primeros permitiéndoles ofrecer un mayor número de servicios en la red, de forma estandarizada, rápida y eficiente en su despliegue, y a los segundos facilitándoles un acceso a servicios de forma inmediata, configurados y de forma transparente. Por el contrario, esta tecnología genera desconfianza y cierto escepticismo en algunos aspectos de seguridad y es además un modelo dependiente del estado de la red.

Las ventajas más importantes que reporta el modelo de computación en la nube son:

- Ahorro generalizado en costes: administración, mantenimiento, monitorización, recuperación ante desastres, políticas de seguridad, etc.
- Gran escalabilidad debido al aprovisionamiento dinámico de recursos.
- Fiabilidad gracias a la presencia de recursos redundantes.
- Abstracción del hardware.
- Sistema de cobro proporcional al consumo.
- Agilidad en la producción y puesta en marcha de servicios.
- Los recursos pueden ser aprovisionados sin coste y de forma rápida.
- Integración garantizada de servicios de red.
- Capacidad de adaptación de los modelos de negocio.
- Efectiva recuperación ante desastres.
- Alta disponibilidad y alto rendimiento.
- Simplicidad en la instalación y mantenimiento por parte del usuario, no precisa de un tipo de hardware específico. Además, proporcionar soporte y actualizaciones resulta más sencillo y directo.
- Reducida inversión inicial para su puesta en marcha por parte del usuario.
- Aprovisionamiento dinámico y rápido de servicios y sus actualizaciones.
- Aumento de la eficiencia energética.
- Independencia de la localización y de los dispositivos para los usuarios de la nube, que sólo precisan acceso Web.

Por el contrario y, como hemos comentado antes, también posee desventajas, casi todas ellas derivadas de la seguridad y la desconfianza que puede generar, son las siguientes:

- Sentimiento de inseguridad o vulnerabilidad ya que el usuario de la nube al no tener el almacenamiento físico de los datos sensibles del negocio a su alcance (para solventarlo las nubes se suelen dotar de eficientes y robustos sistemas de seguridad).
- Dependencia de una conexión a Internet. Debido a la localización en red de los servicios. Se recomienda disponer de conexiones adicionales a usar en caso de fallo de la principal.
- Falta de desarrollo en algunas de las soluciones y servicios Cloud en comparación a los mismos en el modelo clásico de computación.
- Dependencia de los proveedores del servicio al centralizar datos y aplicaciones. La confiabilidad del servicio prestado depende del estado tecnológico y financiero del proveedor. Es posible que servicios altamente especializados no estén disponibles en la red a corto plazo.
- Aparición de posibles amenazas de seguridad debido a la arquitectura multinodo de la nube.
- Disminución de la velocidad si se sobrecarga la red utilizando protocolos para el encriptado de las comunicaciones, por ejemplo.
- Degradación del servicio si la infraestructura de la nube no escala a medida que crece el número de usuarios de la misma.

3.2.3 COMPARACIÓN DE TECNOLOGÍAS

Una vez que se han definido las dos posibles arquitecturas que se van a usar para implementar el sistema servidor, habrá que realizar una comparación para ver cuál se ajusta mejor al sistema que se desea construir. Los apartados que se han seleccionado para la comparación son análisis de rendimiento y almacenamiento de imágenes, ya que son apartados clave a la hora de la realización del proyecto.

Una de las principales ventajas de la opción B es la escalabilidad, además de la autenticación con cuentas de Google para facilitar el acceso y evitar que el usuario tenga que recordar más contraseñas, y delegar el acceso a una compañía confiable y no tener que almacenar contraseñas (con el peligro que ello conlleva).

También es importante el rendimiento, que en la opción B no se ve afectado por la cantidad de información almacenada (que se espera que crezca al aumentar el número de administradores de grupos).

¿Por qué Google App Engine? Dada la relevancia que ha adquirido el paradigma “Cloud Computing” en los últimos años y el prometedor futuro que se le presupone por delante, son muchas las empresas y organizaciones que se han posicionado o intentan hacerlo sobre las demás ofreciendo este tipo de servicios. Existen infinidad de estudios o referencias y reconocidas autoridades en el mundo de las nuevas tecnologías y de los negocios, que predicen un potente mercado entorno a dicho paradigma. El mundo empresarial es cada día más consciente de este hecho, y por ello cada vez son más las soluciones disponibles para los usuarios. En el contexto del proyecto que nos ocupa, se estudiaron las principales plataformas “Cloud Computing” de tipo público. Entre ellas, destacan Google App Engine (GAE), Amazon EC2 y Windows Azure, que proveen aplicaciones comunes en línea accesibles desde un navegador web, mientras el software y los datos se almacenan en los servidores. Todas ellas se basan en el mismo paradigma, pese a que cada una posee sus particularidades y en algunos casos existen diferencias notables entre ellas.

Tras un estudio, finalmente se optó por Google App Engine (GAE), en lugar de Amazon EC2 y Windows Azure, principalmente por un único motivo: su generosa cuota gratuita. Por tratarse de un proyecto universitario de fin carrera, al que no es posible asociar costes ajenos a las horas de trabajo del autor, su tutor y su co-director; el uso de cuotas gratuitas representaba un requisito indispensable. Es por ello por lo que se descartaron Amazon EC2, que no disponía de cuotas gratuitas en el momento del estudio, a pesar de que éstas se han habilitado con posterioridad, y Windows Azure, cuyos recursos gratuitos resultaban insuficientes para el propósito del proyecto.

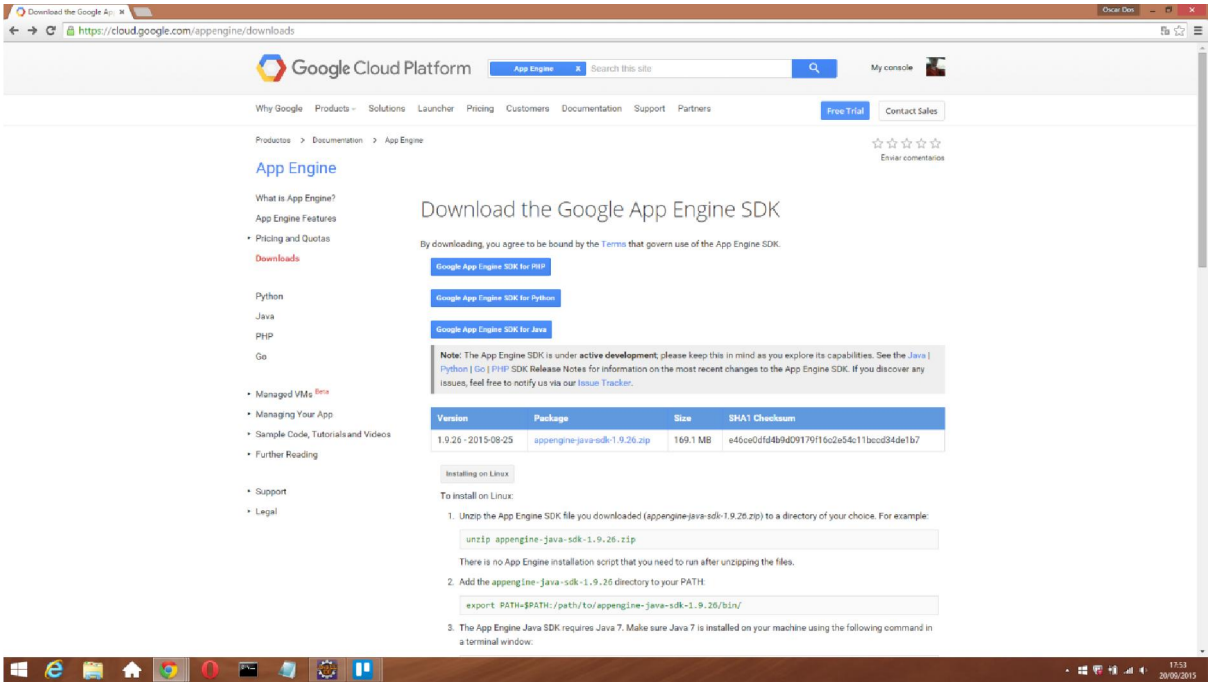


Ilustración 3-5 SDK y Plugin para Eclipse del App Engine 1.

Sobre el plugin de Google para Eclipse, ayuda a los desarrolladores a crear una experiencia de usuario rica; genera alta calidad del código Ajax usando el Google Web Toolkit y despliega aplicaciones de App Engine. El complemento de Google también trabaja con proyectos Apps Script en Drive. Estas herramientas gratuitas para desarrolladores se centran en la creación de una gran lógica de la aplicación. El Google Plugin para Eclipse es la primera suite de herramientas integradas para la nube Google.

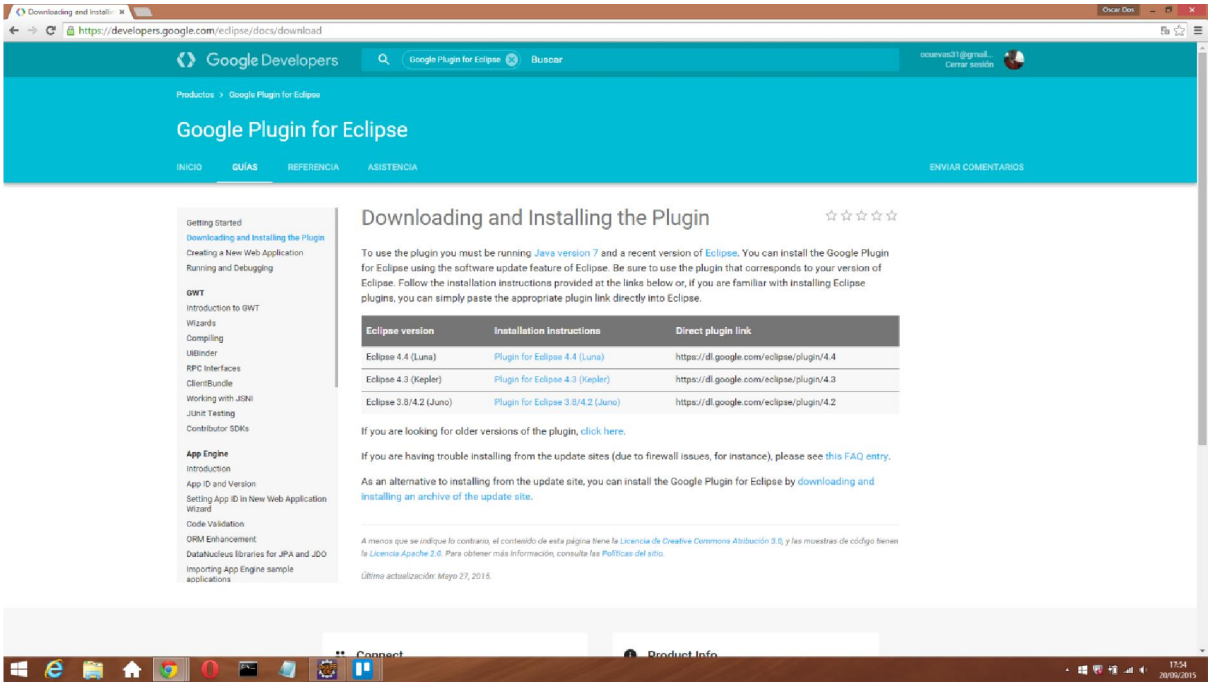


Ilustración 3-6 SDK y Plugin para Eclipse del App Engine 2

EL DATASTORE

Cuando usamos Google App Engine, no tenemos acceso a una base de datos relacional tradicional como MySQL, Oracle o Postgres. Nuestros datos se almacenan en el ***Google Datastore*** que usa un enfoque jerárquico orientado a objetos al estar basado en otra tecnología de Google, el ***Google Bigtable*** que es un sistema distribuido de almacenamiento de datos estructurados.

El enfoque de utilizar Bigtable como almacenamiento a través del Google Datastore consiste en ofrecer una forma eficiente de escalabilidad a nuestras aplicaciones en la nube de Google, las bases de datos NoSQL son conocidas por su predisposición a facilitar la escalabilidad.

Annual Cloud Storage Cost (rounded)					
	FREE	100GB	200GB	500GB (Google – 600GB)	1TB
Google	*15GB*	$24	$48	$72	$120
Microsoft	*7GB*	$50	$100	N/A	N/A
Dropbox	*2GB*	$100	$200	$500	$795

Ilustración 3-7 Coste Almacenamiento en la nube

Por los motivos que se han descrito anteriormente, el sistema compuesto por la Arquitectura B es la opción elegida para la implementación del sistema servidor.

Otra de las secuelas positivas que se ha producido ha sido la aparición del paradigma de computación Cloud Computing. Este tipo de sistemas combina en un sistema de computación único equipos que no tienen por qué estar físicamente en la misma ubicación e incluso no tienen que ser iguales a nivel hardware, integrando todos sus recursos como si de un solo sistema se tratase. Beneficiándose de las comunicaciones de red, se crea un sistema en el que varios servidores trabajan de manera conjunta y aprovechan todos sus recursos poniéndolos a disposición del entorno global.

Los sistemas Cloud Computing ofrecen a los usuarios a través de la red recursos, sistemas completos o servicios que de otra forma serían muy costosos de implantar, tanto por costes hardware como de trabajo de configuración e implantación. Estos sistemas permiten al usuario abstraerse del hardware y trabajar directamente sobre el entorno, aprovechando las bondades de la virtualización de servidores. La idea principal de estos entornos de servidores que forman un Cloud, es poder desplegar instancias virtuales que permitan aprovechar al máximo los recursos.

Una de las principales ventajas es que no se trabaja directamente sobre los servidores y puede crear y eliminar instancias virtuales sin preocuparse sobre la compatibilidad o no del hardware y los drivers del sistema. El objetivo principal de los entornos Cloud Computing es dotar al usuario de servicios como infraestructuras informáticas completas, plataformas para el desarrollo de aplicaciones o software, de forma general. Ésta es la prestación de la

empresa proveedora del Cloud a sus usuarios, siendo para estos últimos indiferente el hardware. Además, dicha empresa es la encargada del establecimiento y el mantenimiento de la infraestructura que da soporte a sus servicios. Se trata de una tendencia de computación cada vez más fiable, con una escalabilidad elástica capaz de atender exigentes cambios en la demanda, sin que ello suponga incrementos en los costes de administración y gestión. Su introducción ha permitido definir un nuevo modelo por consumo de servicios, adaptando nuevas formas para el cobro por recursos consumidos.

3.3 DISEÑO

3.3.1 PATRONES DE DISEÑO:

Los patrones de diseño son soluciones para problemas típicos y recurrentes que nos podemos encontrar a la hora de desarrollar una aplicación.

Aunque nuestra aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas, etc. En lugar de reinventar la rueda, podemos solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores.

CATÁLOGO DE 23 PATRONES DE DISEÑO:

Abstract Factory: tiene como objetivo la creación de objetos reagrupados en familias sin tener que conocer las clases concretas destinadas a la creación de estos objetos.

Builder: permite separar la construcción de objetos complejos de su implementación de modo que un cliente pueda crear estos objetos complejos con implementaciones diferentes.

Factory Method: tiene como objetivo presentar un método abstracto para la creación de un objeto reportando a las subclases concretas la creación efectiva.

Prototype: permite crear nuevos objetos por duplicación de objetos existentes llamados prototipos que disponen de la capacidad de clonación.

Singleton: permite asegurar que de una clase concreta existe una única instancia y proporciona un método único que la devuelve.

Adapter: tiene como objetivo convertir la interfaz de una clase existente en la interfaz esperada por los clientes también existentes para que puedan trabajar de forma conjunta.

Bridge: tiene como objetivo separar los aspectos conceptuales de una jerarquía de clases de su implementación.

Composite: proporciona un marco de diseño de una composición de objetos con una profundidad de composición variable, basando el diseño en un árbol.

Decorator: permite agregar dinámicamente funcionalidades suplementarias a un objeto.

Facade: tiene como objetivo reagrupar las interfaces de un conjunto de objetos en una interfaz unificada que resulte más fácil de utilizar.

Flyweight: facilita la compartición de un conjunto importante de objetos con granularidad muy fina.

Proxy: construye un objeto que se sustituye por otro objeto y que controla su acceso.

Chain of responsibility: crea una cadena de objetos tal que si un objeto de la cadena no puede responder a una petición, la pueda transmitir a sus sucesores hasta que uno de ellos responda.

Command: tiene como objetivo transformar una consulta en un objeto, facilitando operaciones como la anulación, la actualización de consultas y su seguimiento.

Interpreter: proporciona un marco para dar una representación mediante objetos de la gramática de un lenguaje, con el objetivo de evaluar, interpretándolas, expresiones escritas en este lenguaje.

Iterator: proporciona un acceso secuencial a una colección de objetos sin que los clientes se preocupen de la implementación de esta colección.

Mediator: construye un objeto cuya vocación es la gestión y el control de las interacciones en el seno de un conjunto de objetos sin que estos elementos se conozcan mutuamente.

Memento: salvaguarda y restaura el estado de un objeto.

Observer: construye una dependencia entre un sujeto y sus observadores de modo que cada modificación del sujeto sea notificada a los observadores para que puedan actualizar su estado.

State: permite a un objeto adaptar su comportamiento en función de su estado interno.

Strategy: adapta el comportamiento y los algoritmos de un objeto en función de una necesidad concreta sin por ello cargar las interacciones con los clientes de este objeto.

Template Method: permite reportar en las subclases ciertas etapas de una de las operaciones de un objeto, estando éstas descritas en las subclases.

Visitor: construye una operación a realizar en los elementos de un conjunto de objetos. Es posible agregar nuevas operaciones sin modificar las clases de estos objetos.

3.3.2 MVC

El MVC (Modelo-Vista-Controlador) es un patrón de diseño muy utilizado en aplicaciones Web, existen muchos frameworks en J2EE que se hubiesen podido utilizar para el desarrollo del prototipo. El patrón de diseño MVC separa en tres capas lógicas la interfaz gráfica, de los datos y la lógica de control.

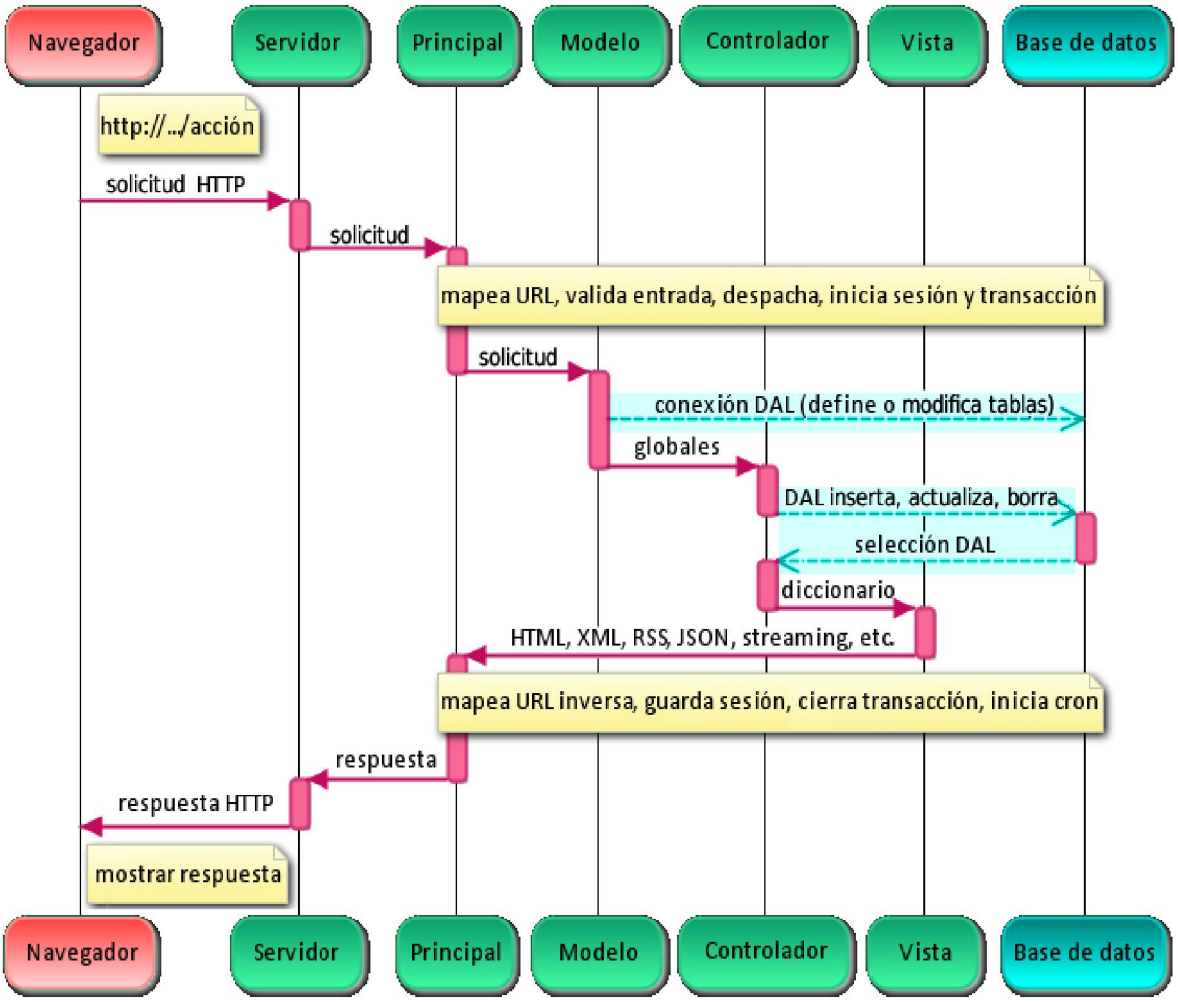


Tabla 3-1 Flujo de Trabajo en el MVC[xxxvii]

3.3.2.1 MODELO:

Es donde se gestiona la lógica de la aplicación Web, y donde se desarrollan todos los accesos al almacén de datos, por ello normalmente existe una réplica en objeto de cada tabla de la base de datos. Al trabajar con Google Application Engine no disponemos de una base de datos relacional, con lo cual también tendremos un objeto por cada tabla del almacén de datos

y se controlarán mediante el código y la lógica todas las restricciones de la aplicación. Para complementar esta capa, se tendrá una clase que será la responsable de trabajar con el almacén de datos, permitiendo, crear, modificar, consultar y eliminar cualquier dato del almacén de datos. Para la gestión de la lógica de la aplicación Web se utilizará el patrón de diseño fachada, para proporcionar en un controlador una interfaz unificada de las funcionalidades que hará de intermediaria entre la capa controlador y modelo en MVC y que podrá ser llamada en ocasiones puntuales por la Vista, el patrón de diseño fachada es poco cohesivo, es decir concentra en una clase todas las responsabilidades, por ello, en algunos casos se dividirá la responsabilidad en un nuevo controlador que agrupe varias responsabilidades comunes.

3.3.2.2 VISTA:

La vista es la responsable de generar la página HTML que se mostrará al usuario, en el informe del proyecto se habló de un diseño orientado a workflows (flujos de trabajo), aunque existen muchas metodologías y definiciones de workflows, es en la interfaz en la que tendremos en cuenta un diseño pensado en las opciones que puede hacer el usuario, y en el hecho de que todo lo que no aparezca en la interfaz no se podrá hacer.

Para diseñar la vista, como podremos ver unos puntos más adelante en el diseño de la interfaz, se ha desarrollado de manera que existe una cabecera, un contenido y un pie, separando la cabecera y el pie en archivos separados, fomentando así un diseño más eficiente en cuanto a mantenimiento y reutilización, por lo tanto en cada página se incluirán los distintos archivos y se generará el contenido, en la cabecera también se incluirá según el tipo de usuario el menú de funcionalidades a mostrar.

También el diseño de las páginas está hecho con hojas de estilo CSS, que permite en un sólo archivo o varios, gestionar todo el estilo de la aplicación, como por ejemplo, la tipografía, el fondo, etc.

Existen varias clases que dan soporte para hacer un diseño aun más reutilizable y de fácil mantenimiento, por ejemplo tenemos funciones que generan trozos de código que se repiten en varias ocasiones, como el contenido estático de listas desplegables, concentrando en sólo un lugar el contenido y pudiendo añadir, modificar y eliminar elementos en sólo un lugar, afectando el cambio a todas las partes en que es utilizado.

Al diseñar la interfaz del prototipo se ha tenido en cuenta algunos aspectos de accesibilidad, como los que afectan a los equipos y medios que se puedan emplear para visualizar el prototipo de aplicación Web.

3.3.2.3 CONTROLADOR:

Es el encargado de gestionar los eventos que ocurren en la aplicación Web, alternándolos con la vista, puesto que también puede generar contenido. El controlador será el encargado de realizar la mayoría de las validaciones, redirigiendo el flujo a una página de error, en caso de accesos no autorizados o errores producidos en el sistema, y será el encargado de contactar con el modelo para la petición de realización de acciones, recibiendo como respuesta datos de control o los datos solicitados. En J2EE el controlador son los servlets, aunque hay que tener en cuenta que los JSP, a parte de la vista, también, tras el proceso de compilación, dan como resultado servlets.

Un ejemplo de evento es la consecuencia de navegar mediante un botón o un link, que redirige el flujo de trabajo hacia un controlador que recibe el evento, realiza validaciones, como si el usuario tiene permisos, si tiene todos los datos que se necesitan, etc. Realiza la acción indicada y devuelve el control a una vista.

3.3.3 BASE DE DATOS

En sí no es una Base de Datos relacional por lo comentado anteriormente, sino que se usa el concepto de Entidad.

3.3.3.1 ENTIDADES DE LA APLICACIÓN ("TABLAS")

Las Entidades son:

- Grupo (Contiene los datos de los grupos musicales).
- Disco (Discos de los grupos musicales).
- Canción (Canciones de los discos de los grupos).
- CadenaLarga (Letra y TAB de las canciones).
- Frase (Frases célebres de los músicos).
- Noticia (Noticias sobre la cultura musical).
- Registro (Usuarios registrados en el sistema).

En las siguientes tablas se especifican los campos de cada Entidad de la aplicación.

Primero veamos los tipos utilizados:

- El tipo String, es el tipo cadena de caracteres, permite almacenar texto hasta un máximo de 1500 caracteres.
- El tipo List<Tipo> es una lista de elementos del tipo Tipo, internamente puede ser un array o una lista enlazada, en resumen una serie de elementos.

- El tipo Key es el empleado en el Data Store para almacenar las claves primarias, que es un campo que sirve para identificar unívocamente a la Entidad.

En la primera fila de cada tabla aparece el nombre de la Entidad y a continuación los campos de la Entidad: el nombre del campo en la primera columna, el tipo en la segunda y, por último, la descripción del campo.

Tabla 3-2 Entidad Grupo del Data Store

NOMBRE: GRUPO		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
nombre	String	Nombre del grupo
discos	List<Disco>	Lista de discos del grupo
logo	String	Url a la imagen del logo
fini	Date	Fecha de creación
biografía	String	Enlace a la biografía
web	String	Página web oficial
país	String	País origen del grupo
wiki	String	Enlace a Wikipedia
idioma	String	Idioma principal letras
estilo	String	Estilo del grupo
subestilo	String	Sub-estilo del grupo
redes	String	Redes sociales propias
youtube	String	Enlace oficial Youtube
facebook	String	Enlace oficial Facebook
twitter	String	Enlace oficial Twitter
instagram	String	Enlace oficial Instagram
spotify	String	Enlace oficial Spotify
gplus	String	Enlace oficial Google+
tumblr	String	Enlace oficial Tumblr

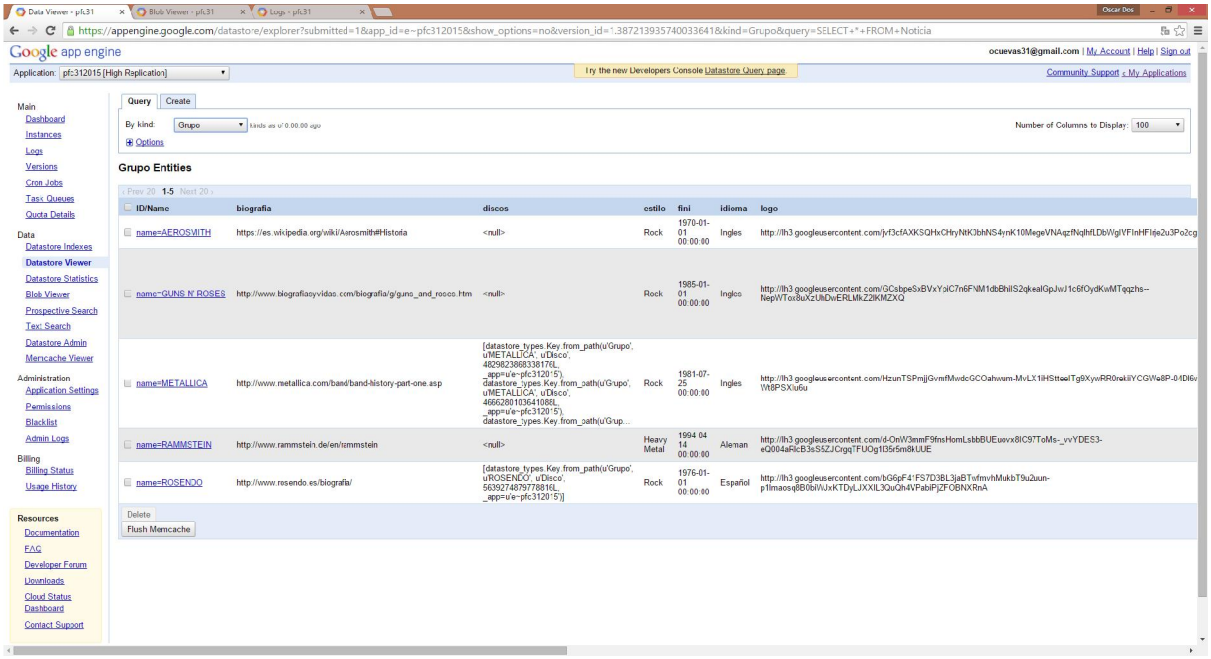


Ilustración 3-8 Ejemplo Datos de la Entidad Grupo del Data Store

Tabla 3-3 Entidad Disco del Data Store

NOMBRE: DISCO		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
key	Key	Clave primaria
canciones	List<Cancion>	Lista de canciones del disco
portada	String	Url a la imagen del disco
titulo	String	Fecha de creación
grupo	String	Nombre del grupo del disco
fecha	Date	Fecha de edición
pais	String	País origen del grupo
web	String	Web oficial
wiki	String	Enlace a Wikipedia
redes	String	Redes sociales propias

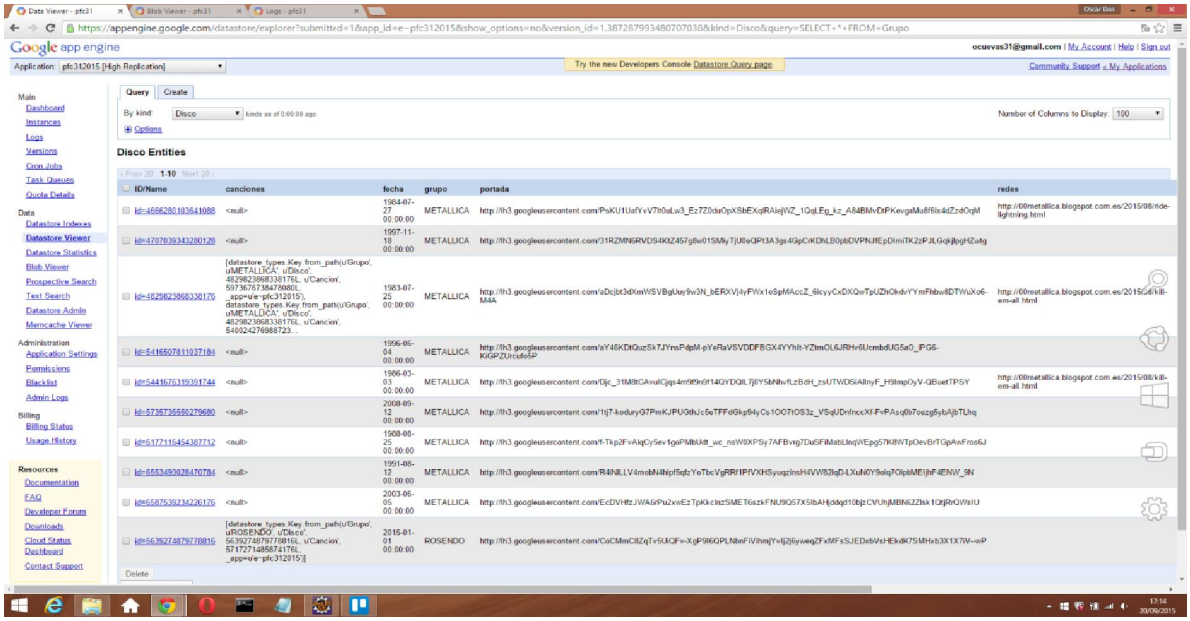


Ilustración 3-9 Ejemplo Datos de la Entidad Disco del Data Store

Tabla 3-4 Entidad Canción del Data Store

NOMBRE: CANCIÓN		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
Key	Key	Clave primaria
tituloCancion	String	Título de la Canción
tituloDisco	String	Título del Disco
letra	CadenaLarga	Letra
traduccion	Cadena Larga	Traducción de la letra
orden	int	Número de orden
resumen	String	Resumen de la letra
mensaje	String	Mensaje de la letra
listaEtiquetas	String	Lista de las etiquetas
tab	CadenaLarga	Tablatura para guitarra
duracion	String	Duración
web	String	Web oficial
wiki	String	Enlace a wikipedia
redes	String	Redes sociales propias
imp	int	Importancia

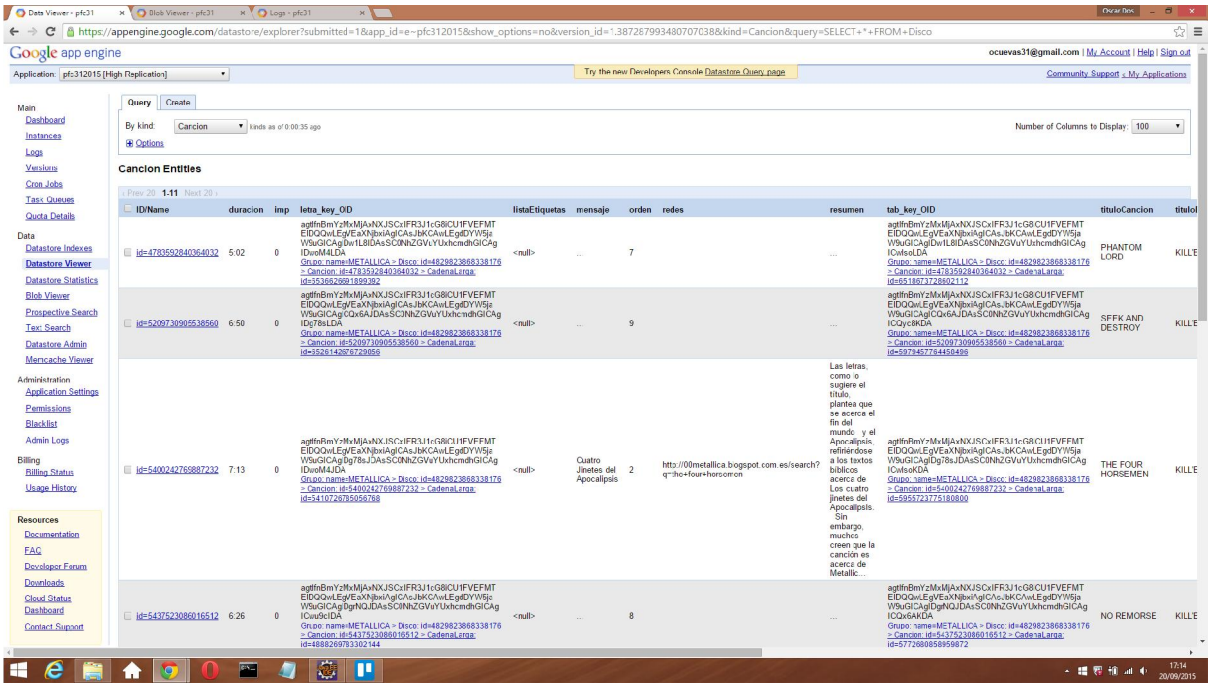


Ilustración 3-10 Ejemplo Datos de la Entidad Canción del Data Store

Tabla 3-5 Entidad CadenaLarga del Data Store

NOMBRE: CADENALARGA		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
Key	Key	Clave primaria
texto	List<String>	Contenido
TAMTROZOS	int	Tamaño de las subcadenas

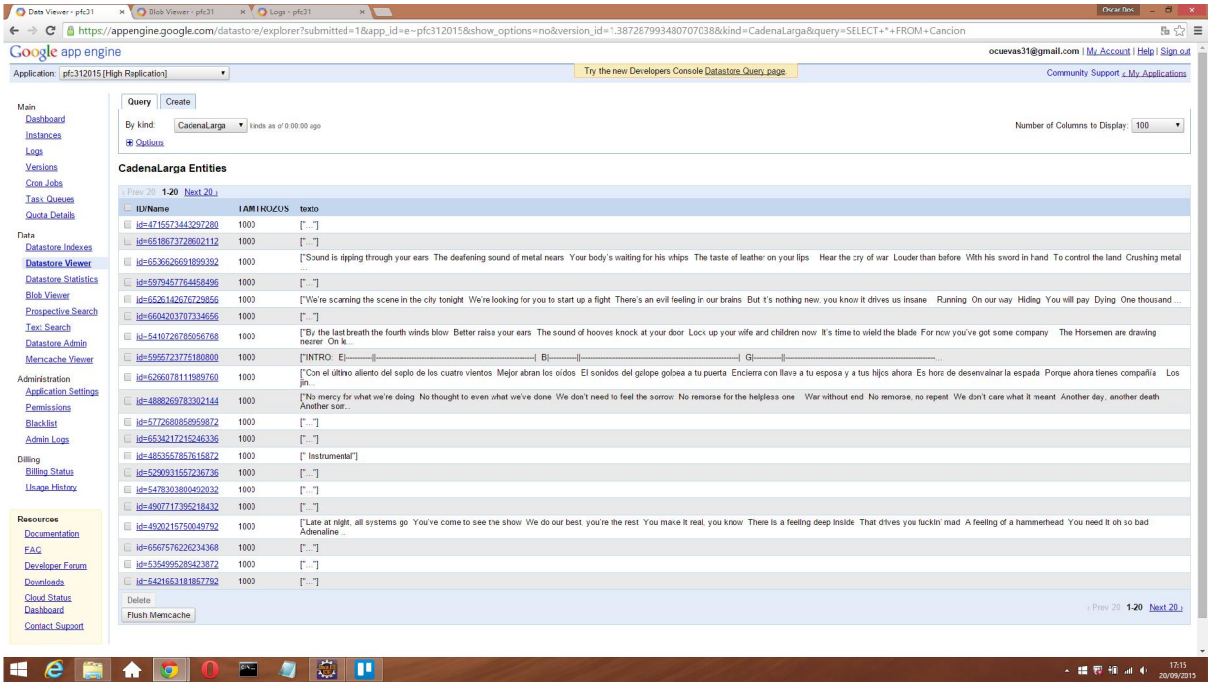


Ilustración 3-11 Ejemplo Datos de la Entidad CadenaLarga del Data Store

Tabla 3-6 Entidad Frase del Data Store

NOMBRE: FRASE		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
Key	Key	Clave primaria
texto	String	Frase célebre

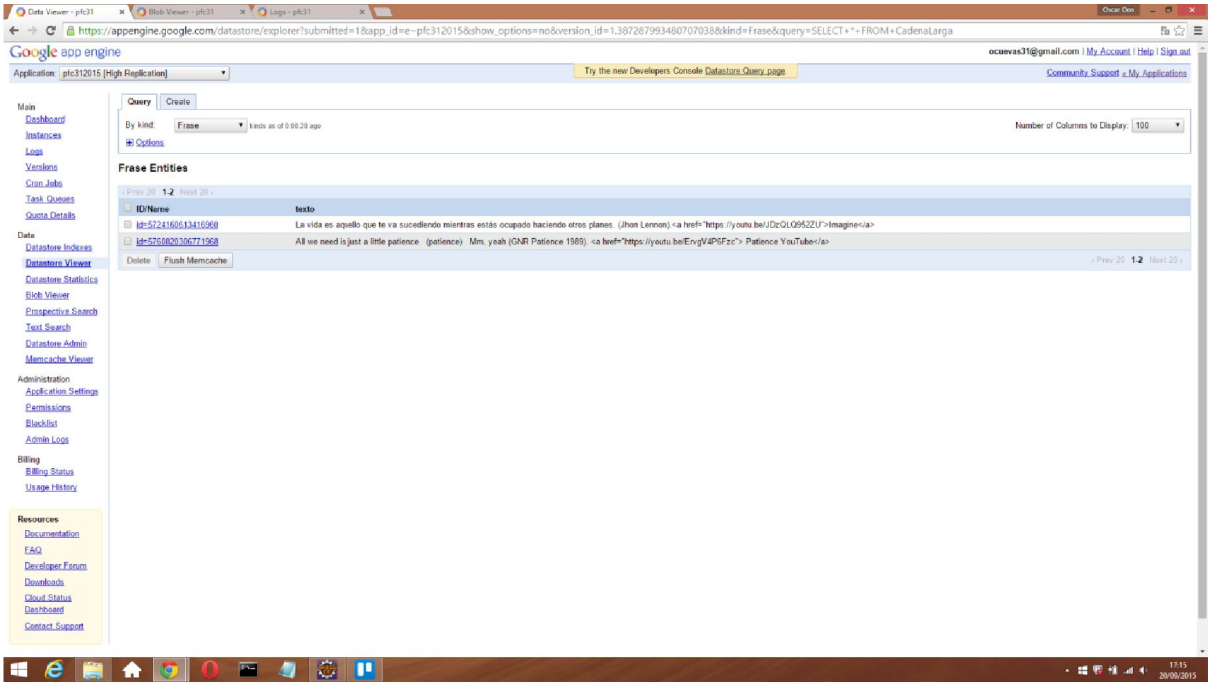


Ilustración 3-12 Ejemplo Datos de la Entidad Frase del Data Store

Tabla 3-7 Entidad Noticia del Data Store

NOMBRE: NOTICIA		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
Key	Key	Clave primaria
texto	String	Noticia

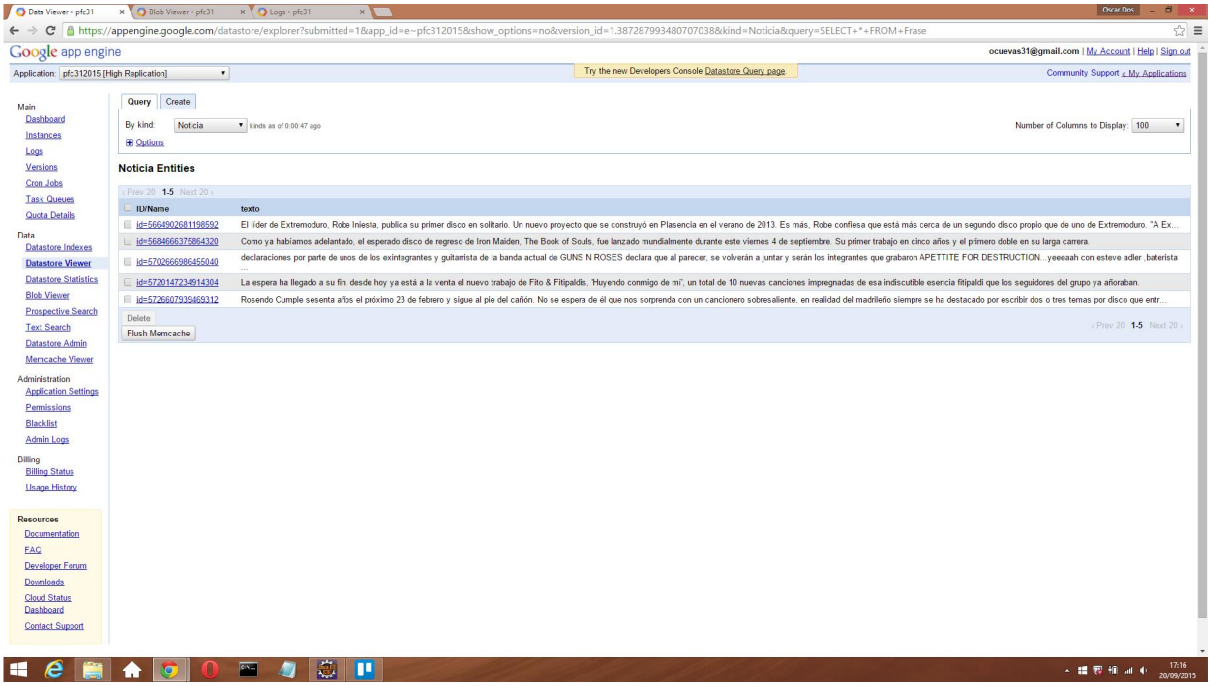


Ilustración 3-13 Ejemplo Datos de la Entidad Noticia del Data Store

Tabla 3-8 Entidad Registro del Data Store

NOMBRE: REGISTRO		
CAMPO:	**TIPO:**	**DESCRIPCIÓN:**
user	String	Clave primaria
grupos	<List>String	Grupos del usuario

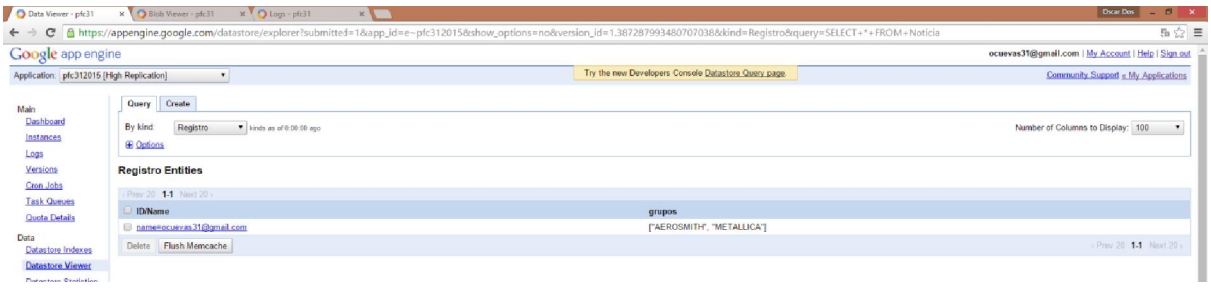


Ilustración 3-14 Ejemplo Datos de la Entidad Registro del Data Store

Los datos binarios, como las imágenes de los logotipos de los grupos y las portadas de los discos se almacenan como Blog Datos, por ejemplo:

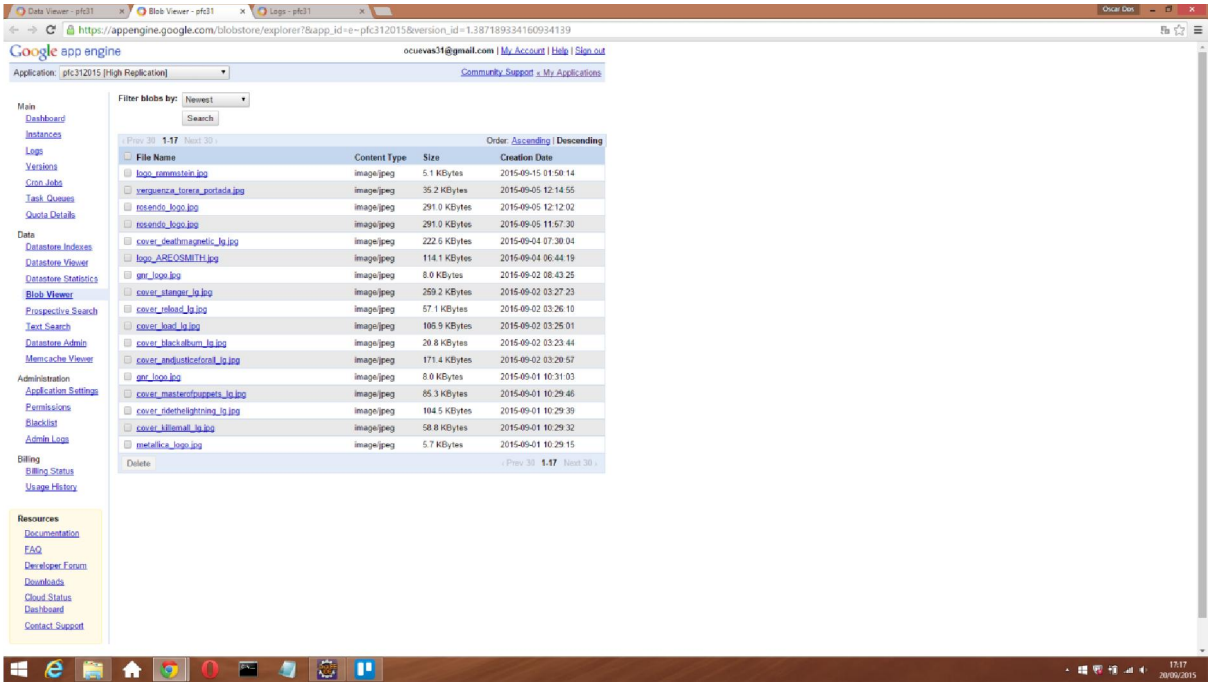


Ilustración 3-15 Ejemplo de Blog Datos del Data Store

Cuotas de almacenamiento y más de un cuenta gratis de GAE:

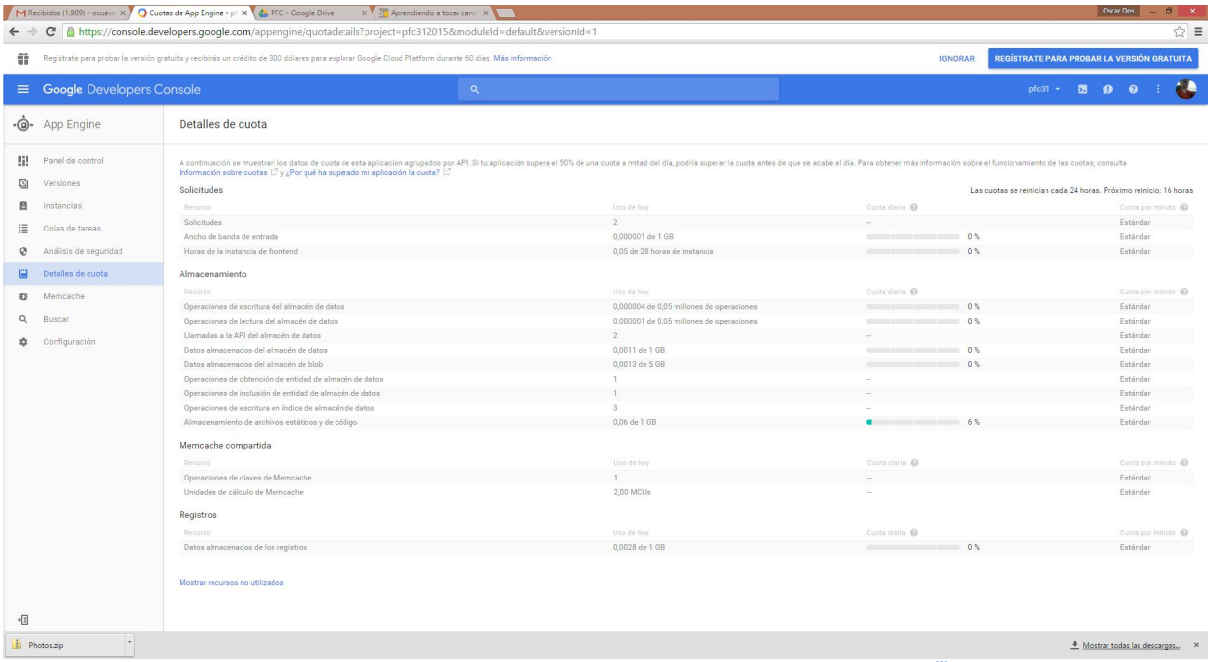


Ilustración 3-16 Detalle de Cuotas en Cuenta Gratis GAE[xxxviii]

Resumen de ocupación desde la consola GAE de las entidades y los índices de la aplicación.

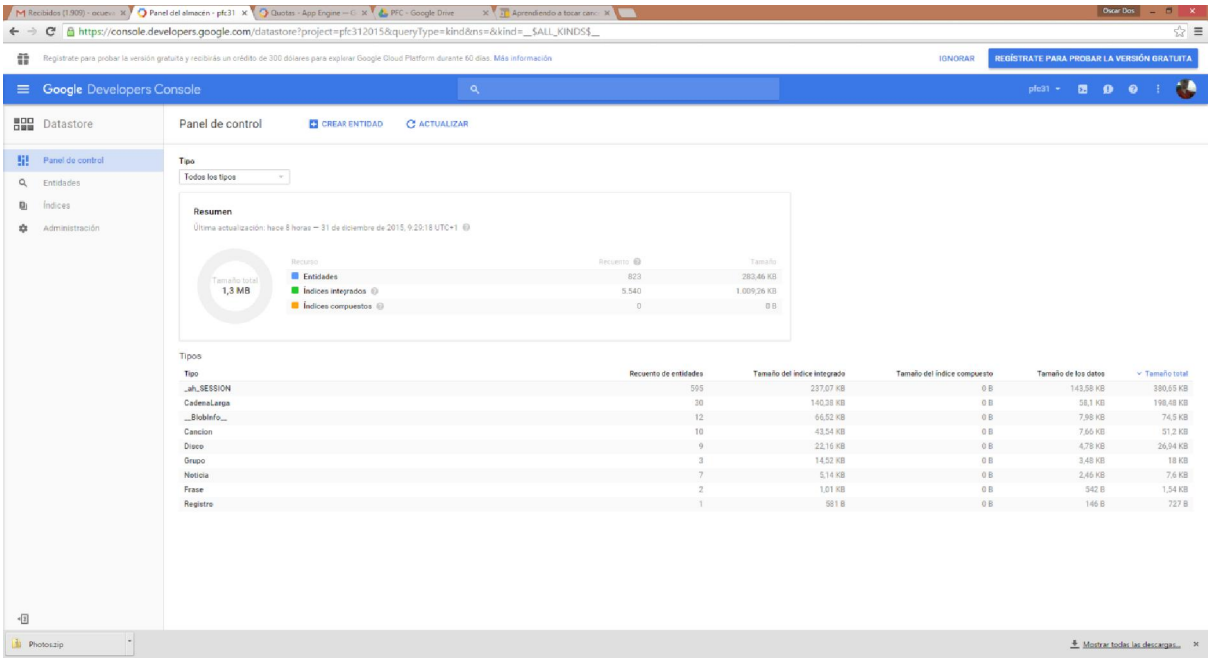


Ilustración 3-17 Resumen Entidades en el Data Store[xxxix].

Veamos ahora algunos logs de la aplicación, gracias a la consola de GAE.

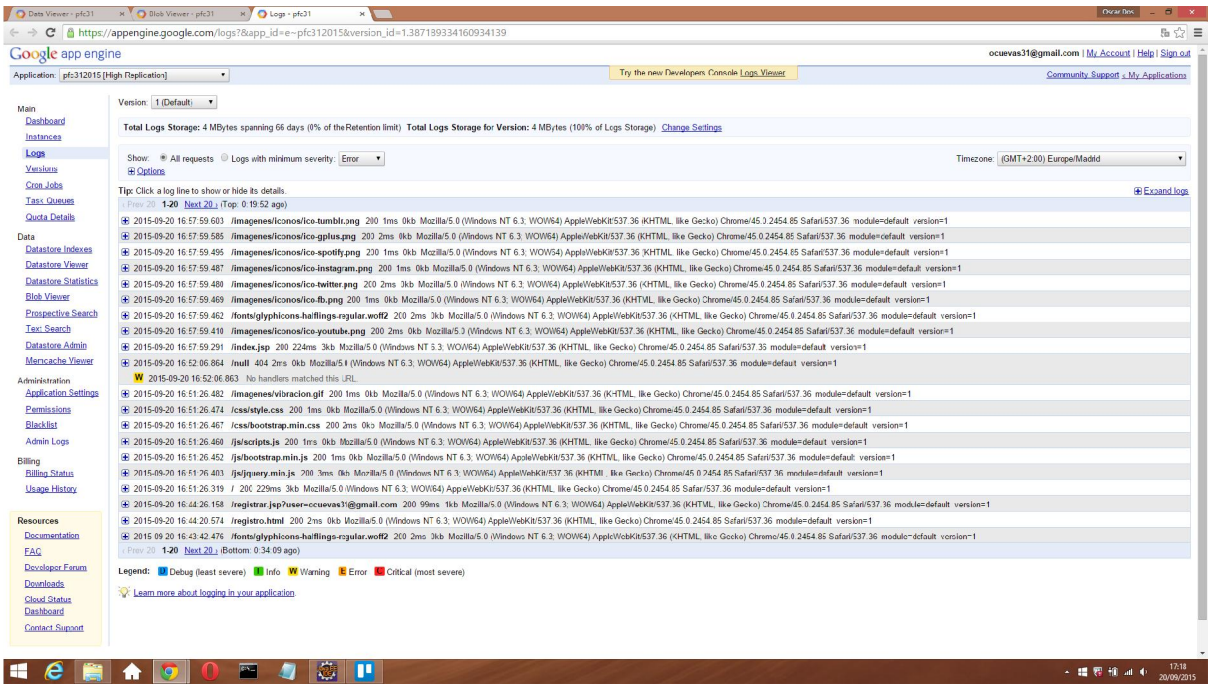


Ilustración 3-18 Logs de la Aplicación desde consola GAE

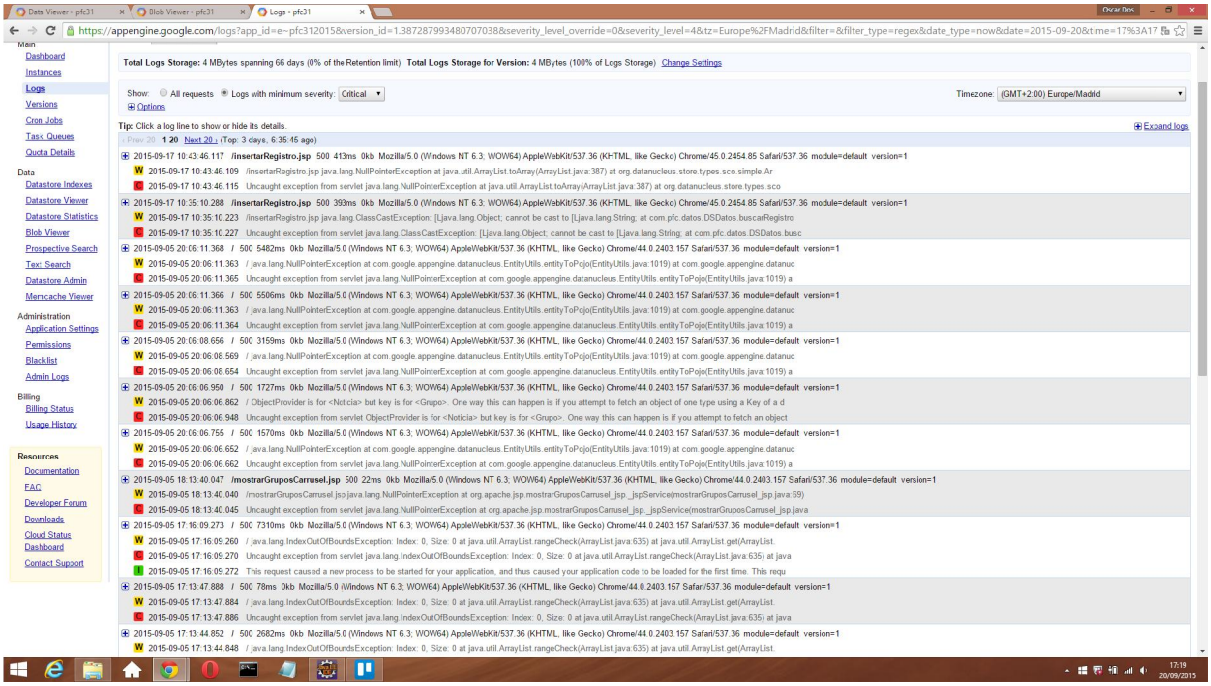


Ilustración 3-19 Logs de Error de la Aplicación desde consola GAE

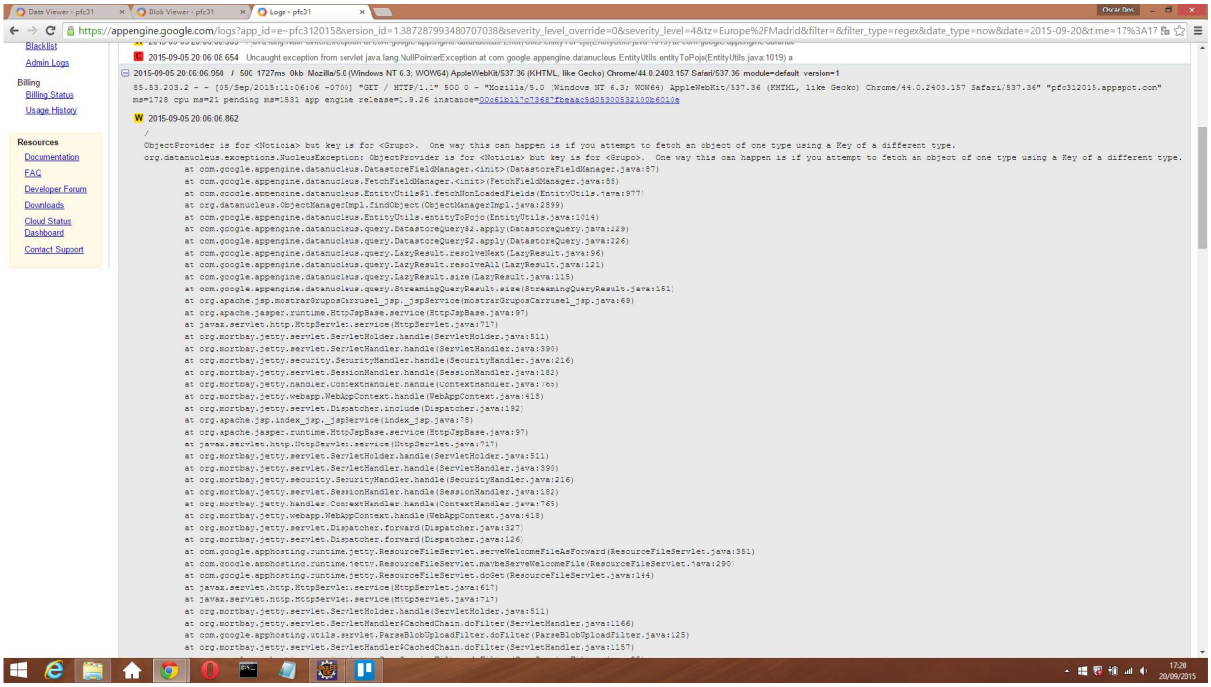


Ilustración 3-20 Logs de Error (detalle) de la Aplicación desde consola GAE

Almacén de DATOS Data Store de Google:

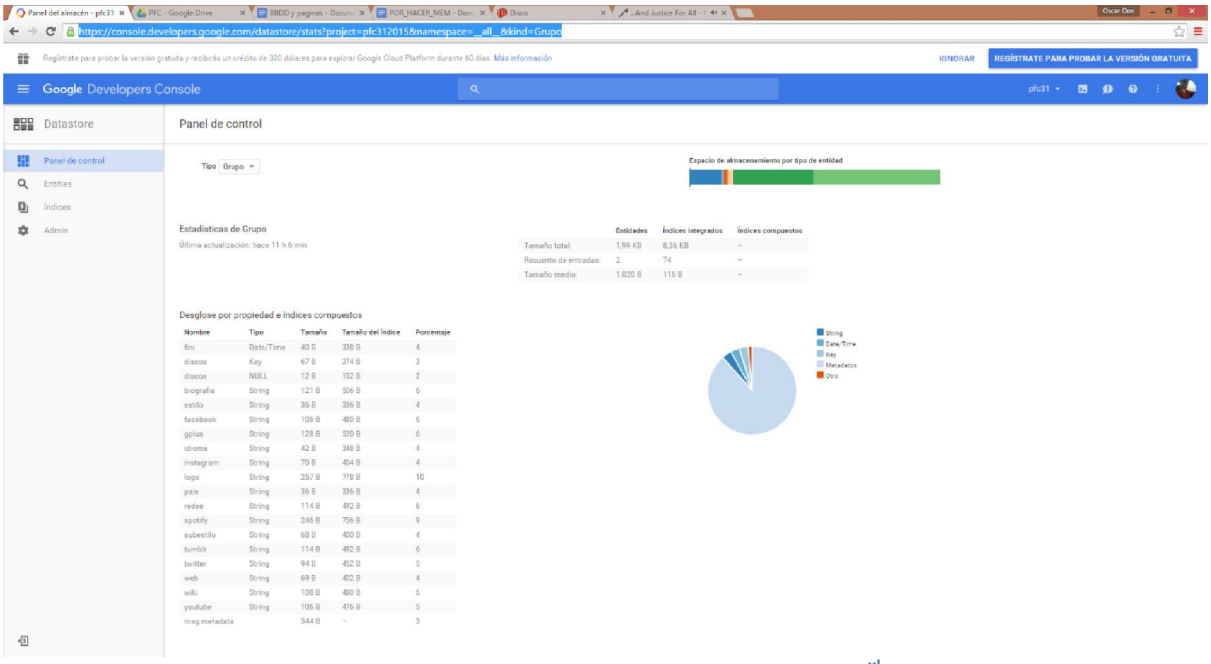


Ilustración 3-21 Entidad Grupo en el Data Store[xl]

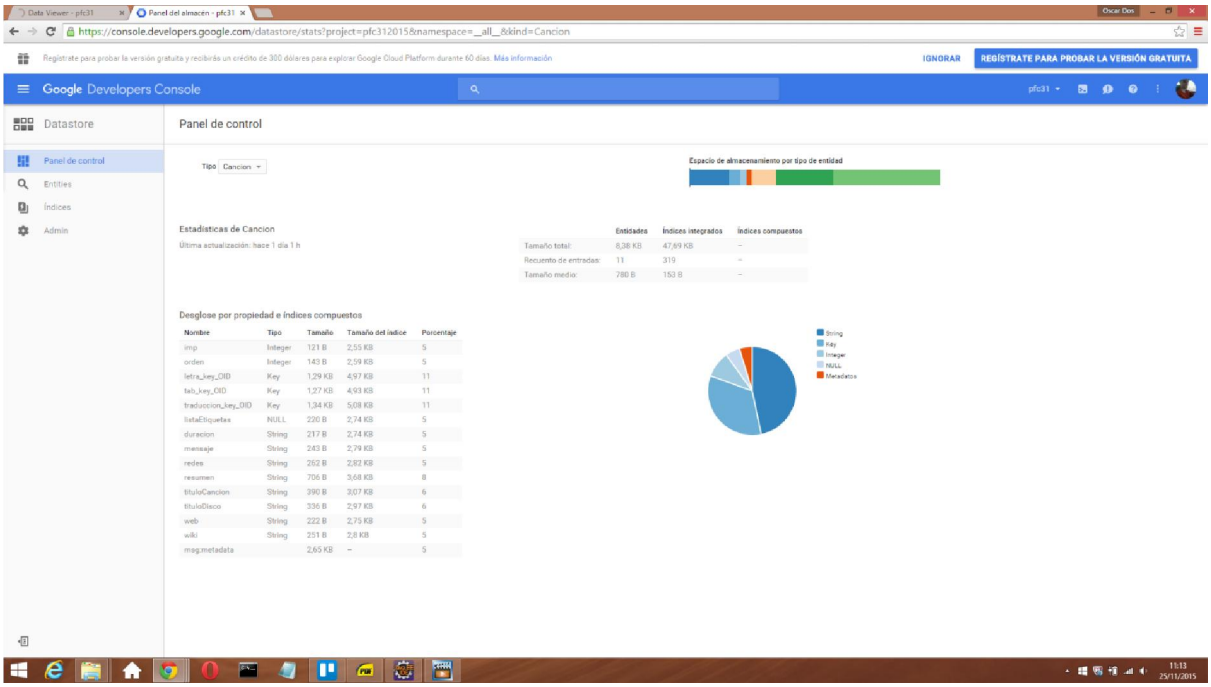


Ilustración 3-22 Entidad Canción en el Data Store

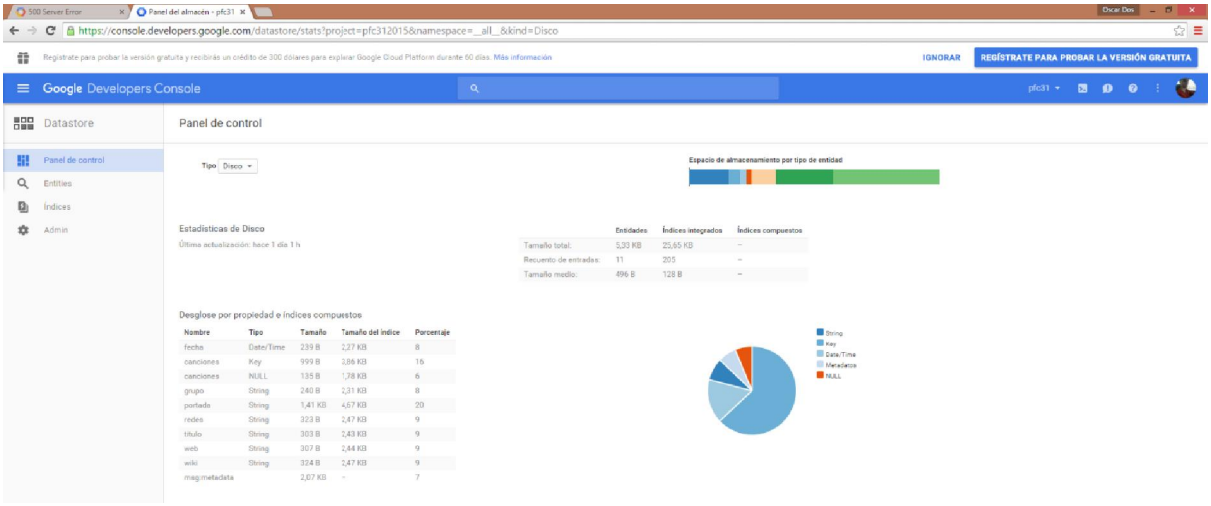


Ilustración 3-23 Entidad Disco en el Data Store

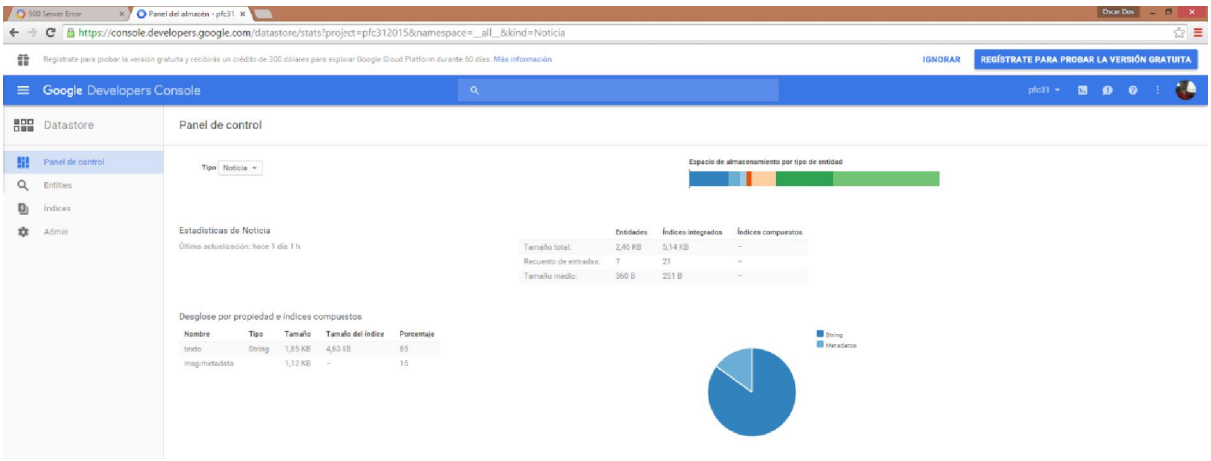


Ilustración 3-24 Entidad Noticia en el Data Store

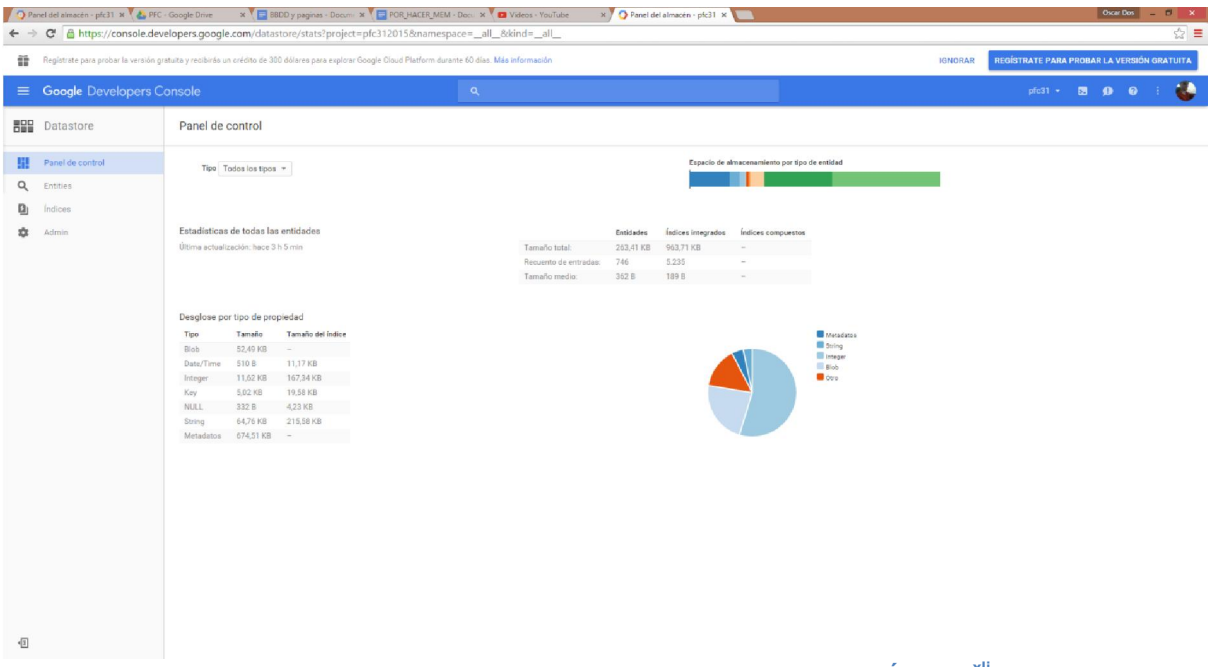


Ilustración 3-25 Tipos de Datos y Tamaños de Entidades e Índices[xli]

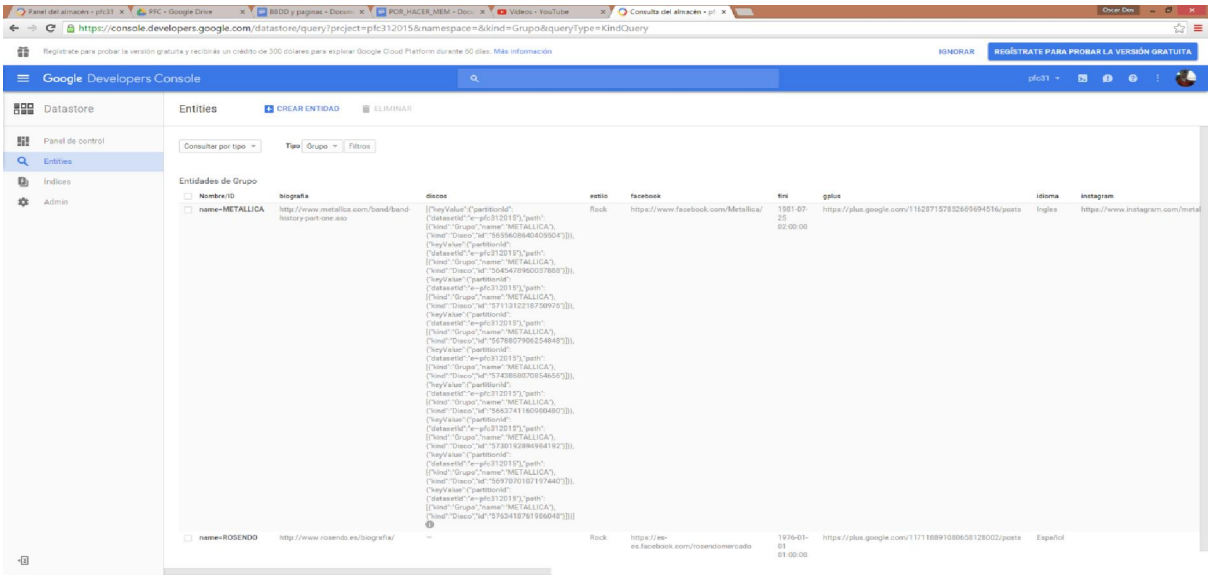


Ilustración 3-26 Ejemplo Entidad Grupo con keys

3.3.3.2 COPIA DE SEGURIDAD (BACKUP)

Una copia de seguridad o copia de respaldo o backup consiste en una copia de los datos originales que se realiza con el fin de disponer de un medio para recuperarlos en caso de su pérdida. Las copias de seguridad son útiles ante distintos eventos y usos: recuperar los sistemas informáticos y los datos de una catástrofe informática, natural o ataque; restaurar una pequeña cantidad de archivos que pueden haberse eliminado accidentalmente, corrompido, infectado por un virus informático u otras causas; guardar información histórica de forma más económica que los discos duros, permitiendo el traslado a ubicaciones distintas de la de los datos originales; etc..

El proceso de copia de seguridad se complementa con otro conocido como **restauración de los datos** (en inglés **restore**), que es la acción de leer y grabar en la ubicación original u otra alternativa los datos requeridos.

La pérdida de datos es muy común, el 66% de los usuarios de Internet han sufrido una seria pérdida de datos en algún momento[xlii]

Con GAE se puede crear una copia de seguridad de los datos del Data Store.

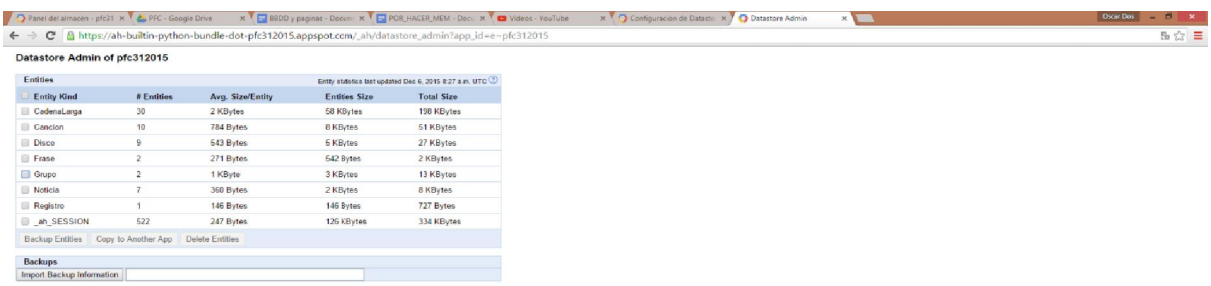


Ilustración 3-27 Copia de Seguridad del Data Store

3.3.3.3 ESTUDIO ANALÍTICO DEL APLICATIVO

En las siguientes ilustraciones se muestra la latencia de algunas de las principales páginas de la aplicación. En general son bastante rápidas, excepto en el momento de subir la aplicación, es decir, en el primer acceso.

Además de la consola de GAE, también se puede utilizar:

https://developers.google.com/speed/pagespeed/insights/
http://tools.pingdom.com/fpt/

Tiempo de carga de la página inicial con una herramienta independiente:
http://tools.pingdom.com/fpt/

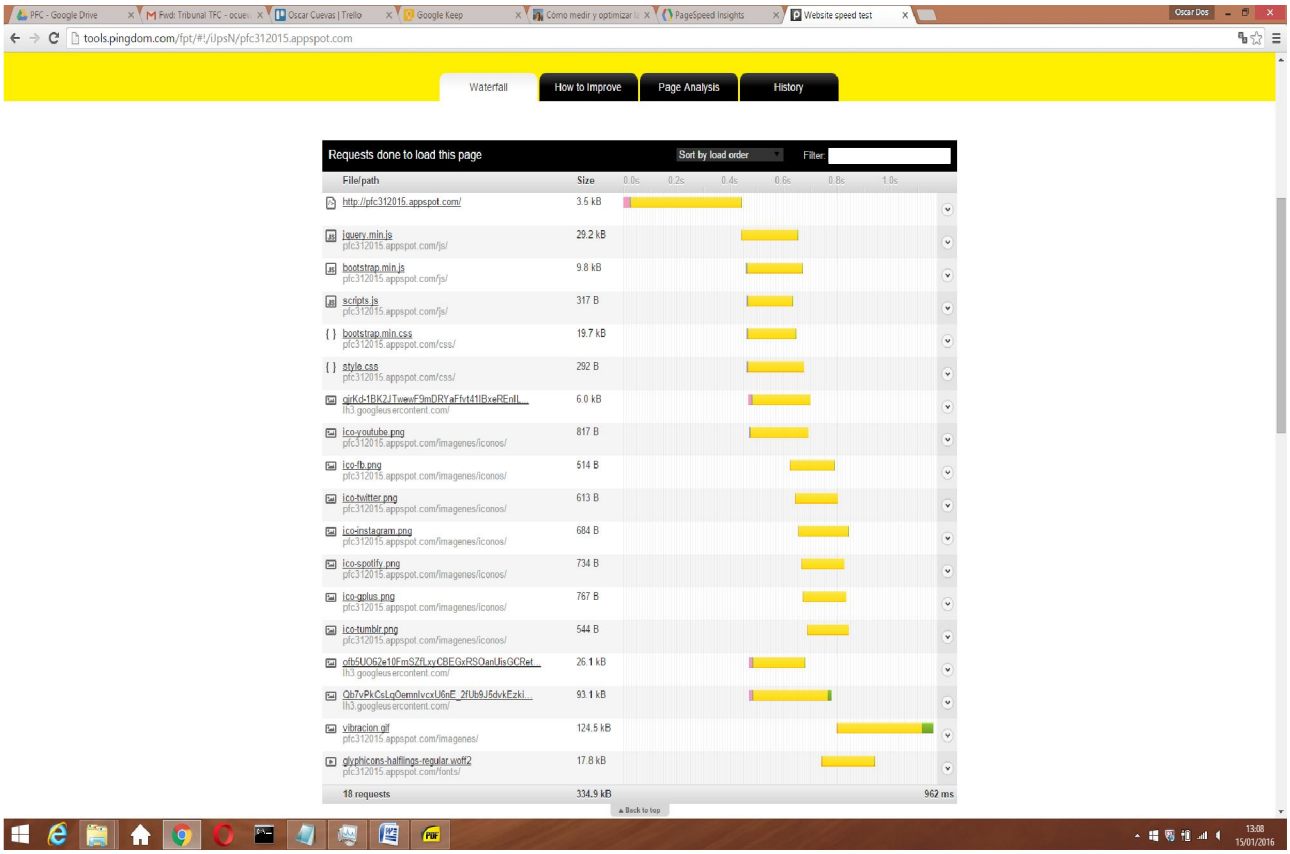


Ilustración 3-28 Análisis de la Velocidad de Página Inicial

Latencia y velocidad de acceso desde consola GAE:

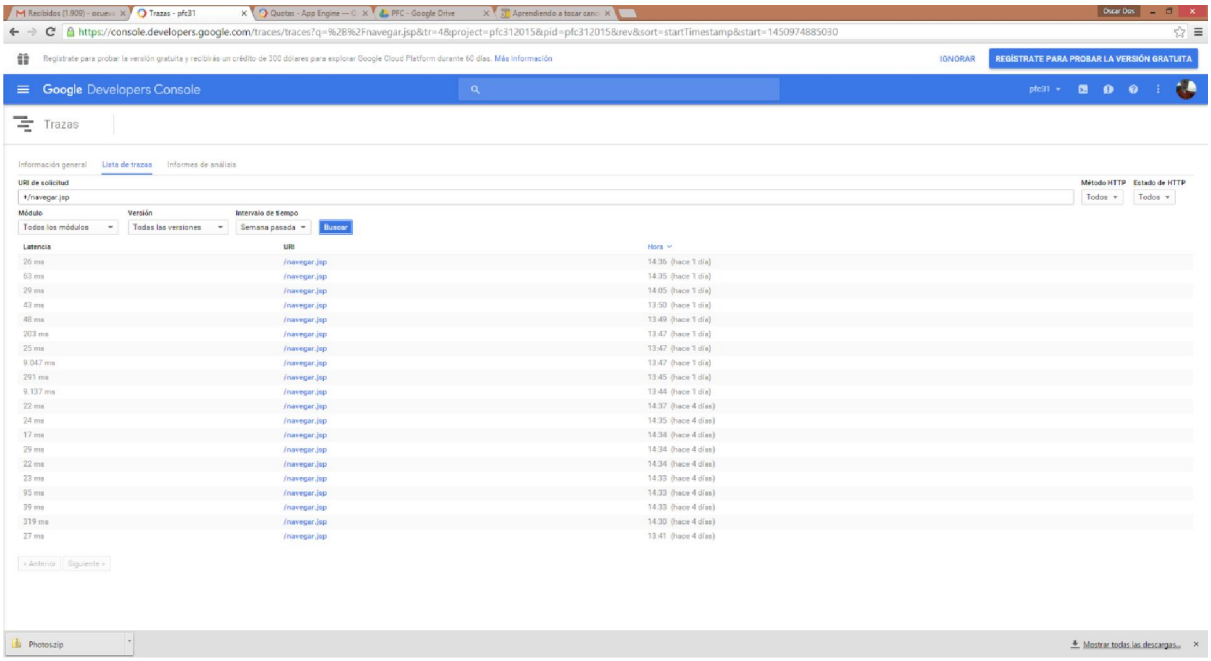


Ilustración 3-29 Latencia y Velocidad de la Aplicación

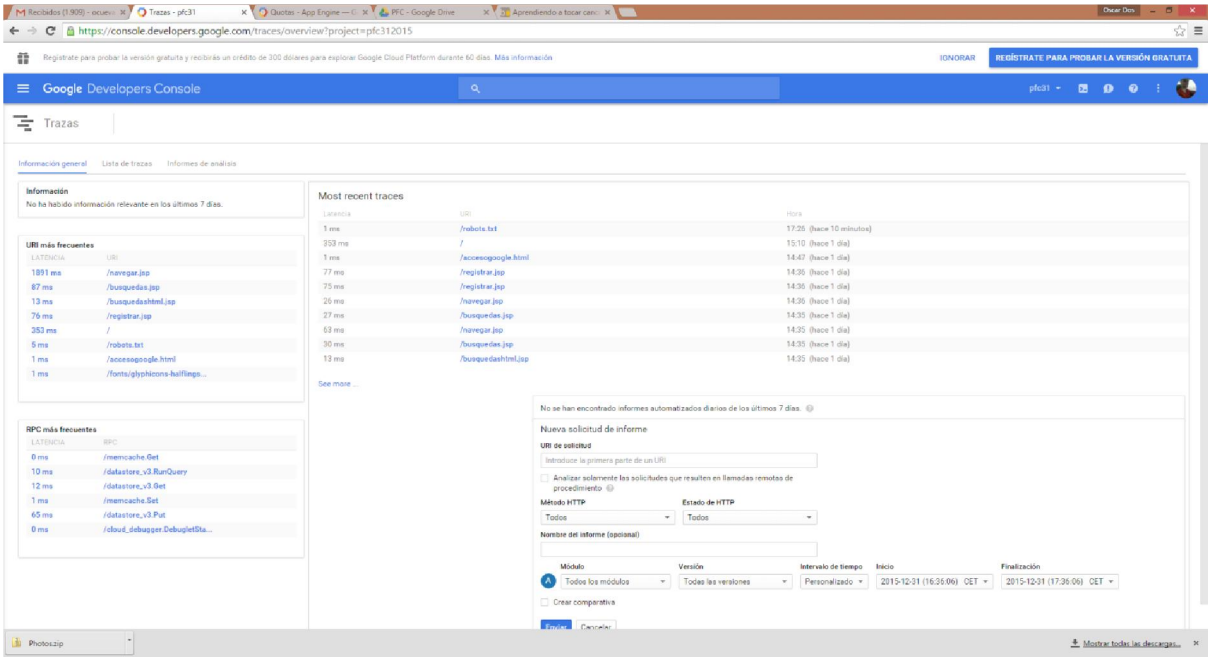


Ilustración 3-30 Latencia de las URI y RPC[xliii] más frecuentes

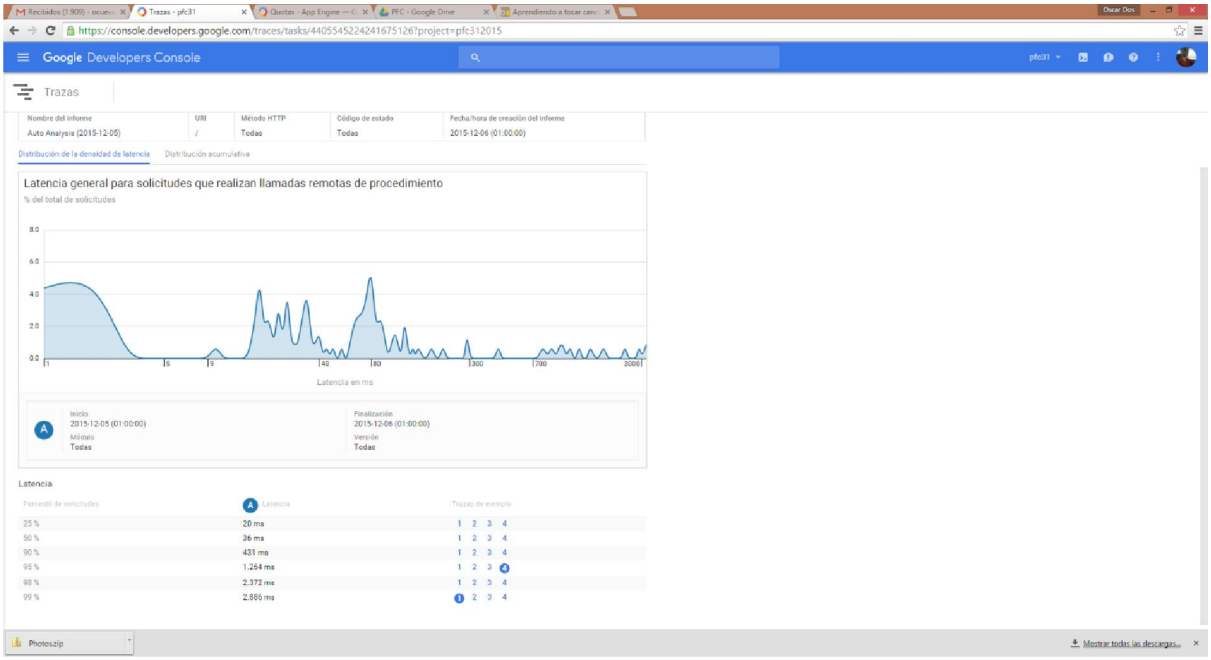


Ilustración 3-31 Latencia General para Solicitudes que realizan RPC

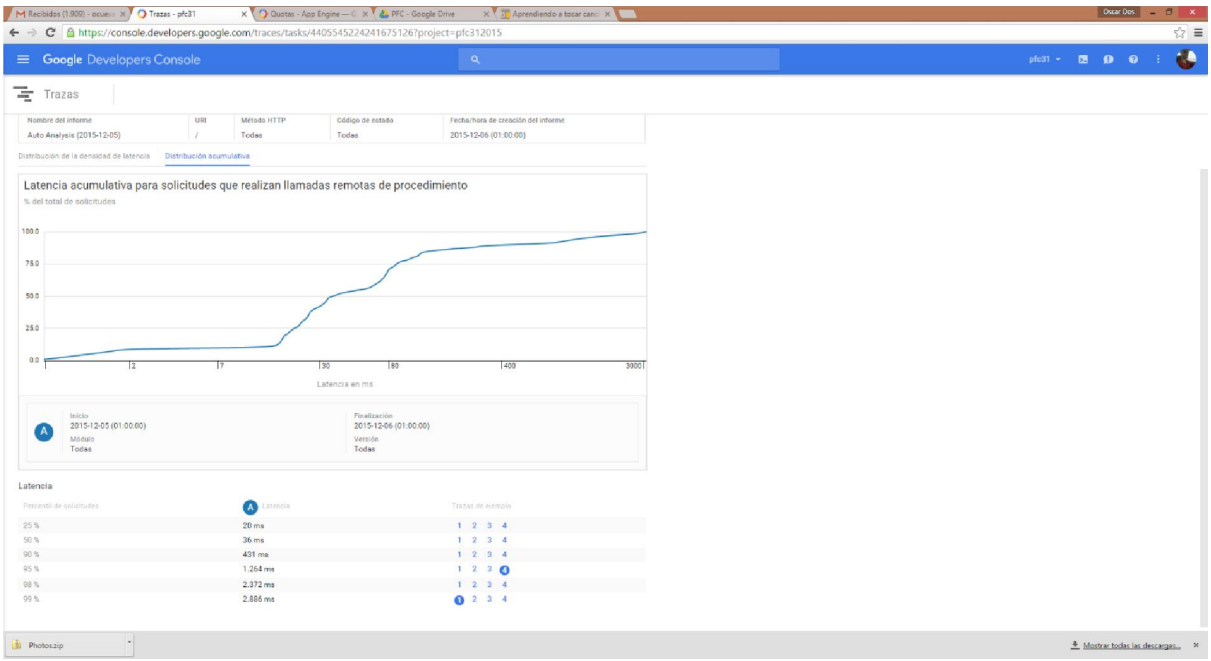


Ilustración 3-32 Latencia Acumulativa para Solicitudes a RPC

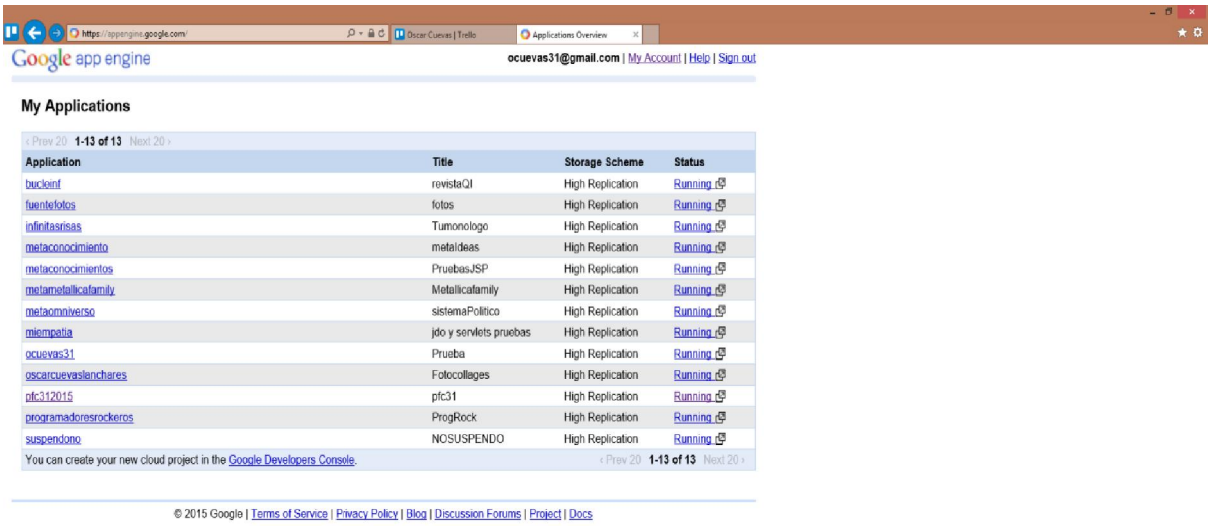


Ilustración 3-33 Listado de la Aplicación del presente PFC y de todas las del autor[xliv]

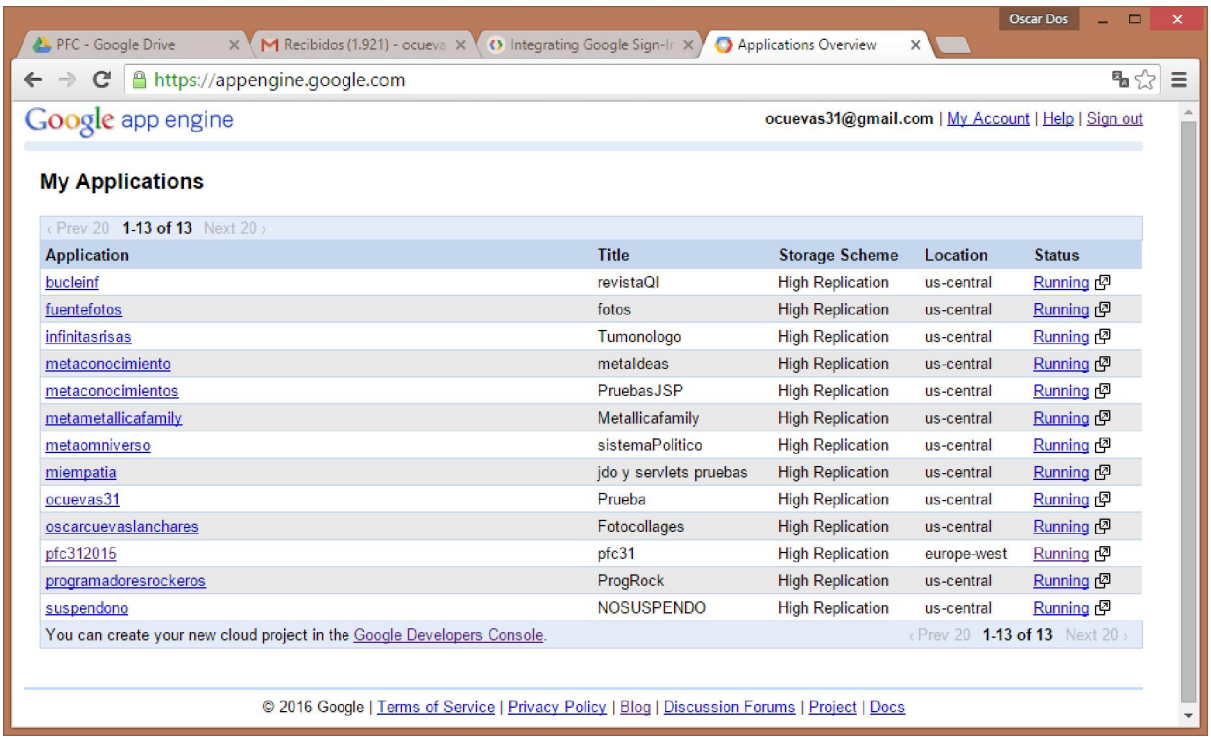
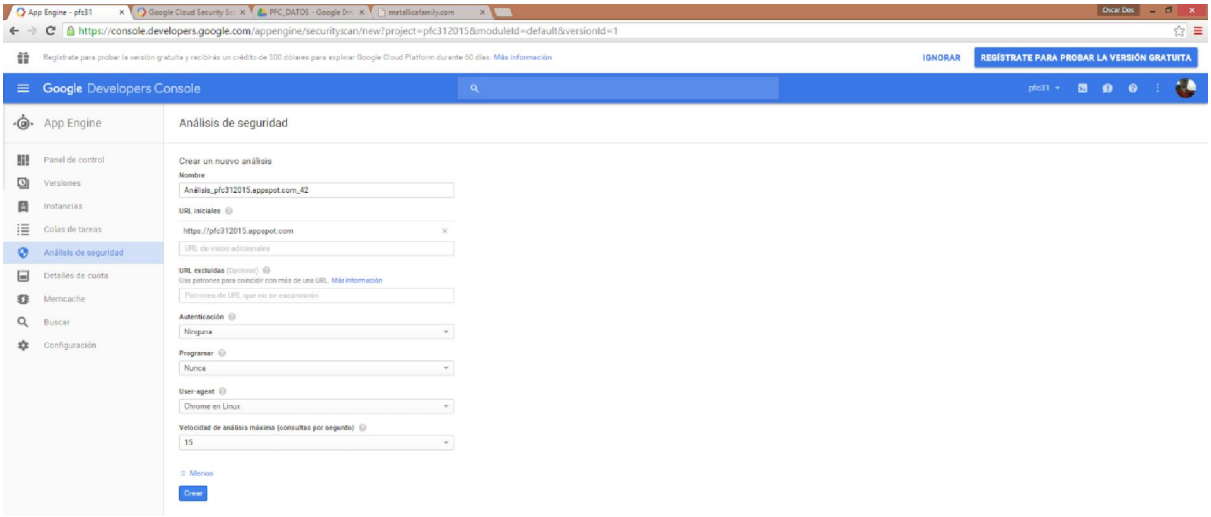
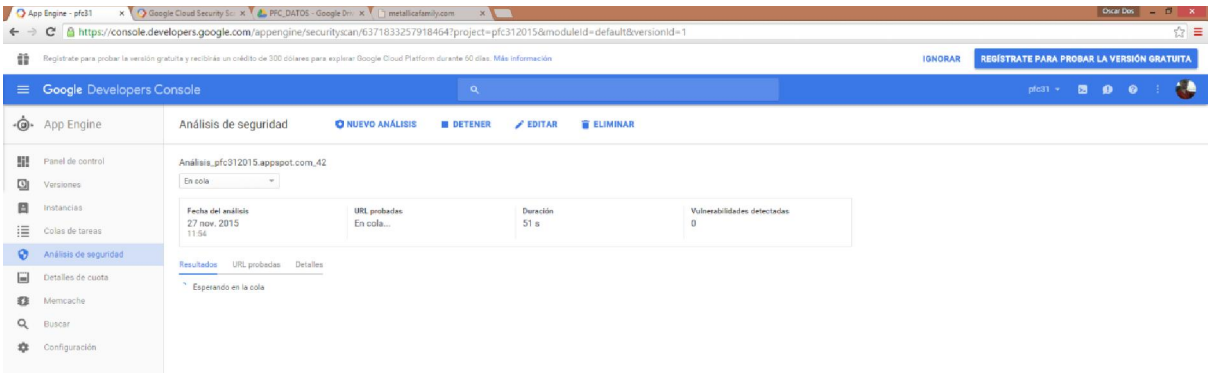


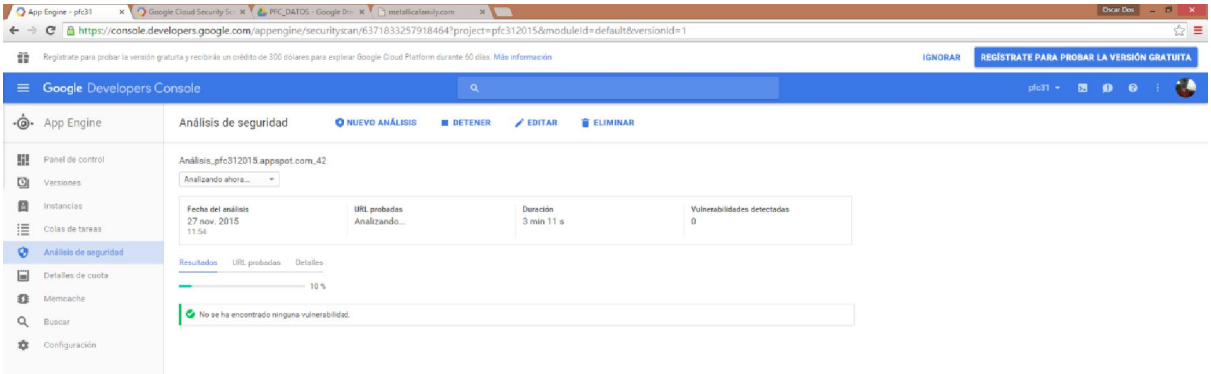
Ilustración 3-34 Nuevo Listado de la Aplicación del presente PFC y de todas las del autor

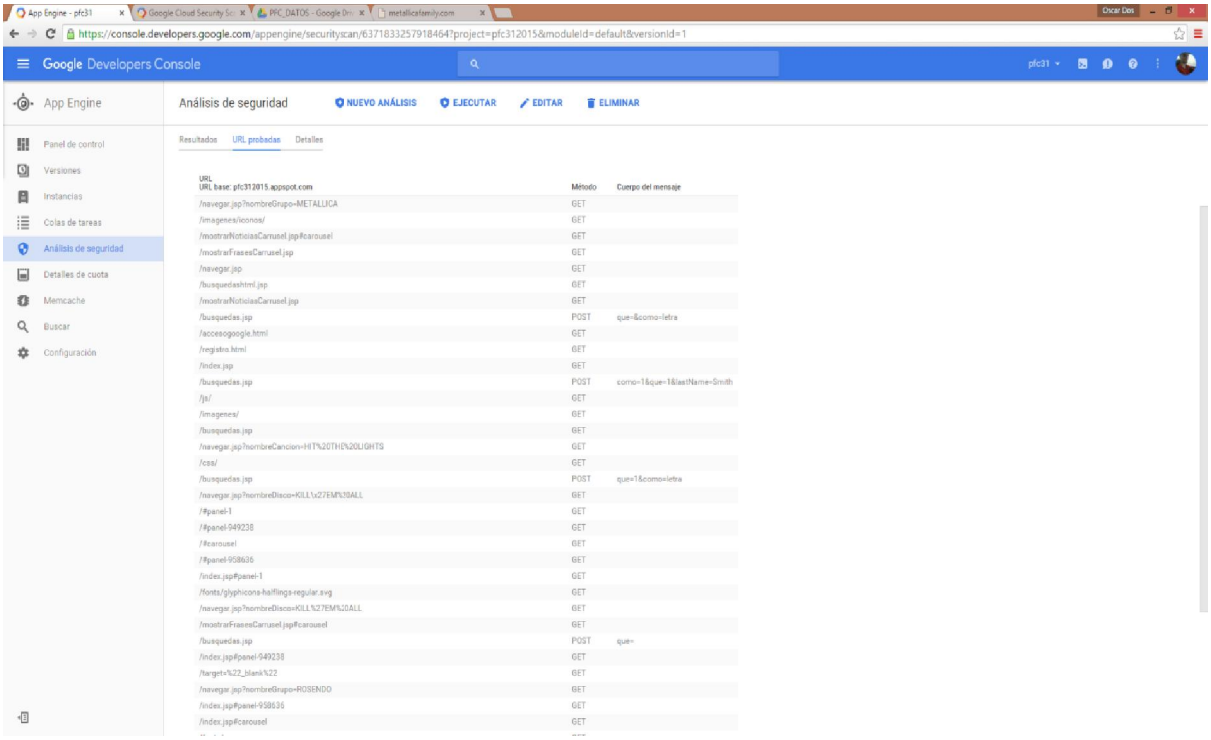
3.3.3.4 ANÁLISIS DE SEGURIDAD

Una posibilidad nueva , que ha aparecido en GAE durante el desarrollo del PFC ha sido realizar un análisis de la seguridad de la aplicación.









3.4 HISTORIAS DE USUARIO

Las historias de usuario muestran la funcionalidad que va a ser desarrollada, pero no cómo se va a desarrollar.

Ron Jeffries[xlv] escribía que una historia de usuario no es sólo una descripción de una funcionalidad, sino que está formada por tres partes:

1. Creación de la tarjeta: debe contener como mínimo una descripción escrita con un título asociado que sirve como identificación de la funcionalidad. También puede contener la estimación y el valor de negocio que tiene la historia para el cliente.

2. Conversación: corresponde con el diálogo que se ha llevado a cabo entre el cliente y el usuario para aclarar los detalles y las dudas que puedan surgir de la historia de usuario.

3. Confirmación: selección entre el equipo del proyecto y el cliente de las pruebas que se realizarán para comprobar que la historia se ha completado con éxito.

Para construir las historias de usuario del proyecto se ha seguido la siguiente estructura:

- **Identificador**: Identificador unívoco de la historia. El identificador aparecerá como título de la tabla y tendrá el formato 'HU-XX', donde HU corresponderá a 'Historia de Usuario y 'XX' al número de la historia.

- **Título**: Título descriptivo de la historia de usuario.
- **Enunciado**: Descripción de la historia de acuerdo con el formato 'Como quiero para' explicado anteriormente.
- **Prioridad**: Valor que aporta la historia de usuario al cliente. Cuanto más peso tenga, más prioridad tendrá. El valor es dado por el cliente con intervención del equipo de desarrollo.
- **Estimación**: Para estimar el tiempo de realización de una historia de usuario, en Scrum se utilizan los puntos de historia. Estos puntos corresponden al esfuerzo que realiza cada miembro del equipo en un día de trabajo. Para determinar los puntos de historia se utilizará la estimación de 'Póquer', que contiene los siguientes valores: 0, ½, 1, 2, 3, 5, 8, 13, 21, ∞ y ?. De esta forma, el valor 1 corresponderá a un día de trabajo, que constará de 4 horas.
- **Dependencias**: Historias de usuario de las que depende la Historia actual.
- **Criterios de Aceptación:** Resultado de realizar la historia de usuario. Se definen con el cliente.
- **Tareas:** División de la historia de usuario.
- **Pruebas de Aceptación:** Pruebas que la historia de usuario debe superar una vez realizada para poder darla como finalizada.

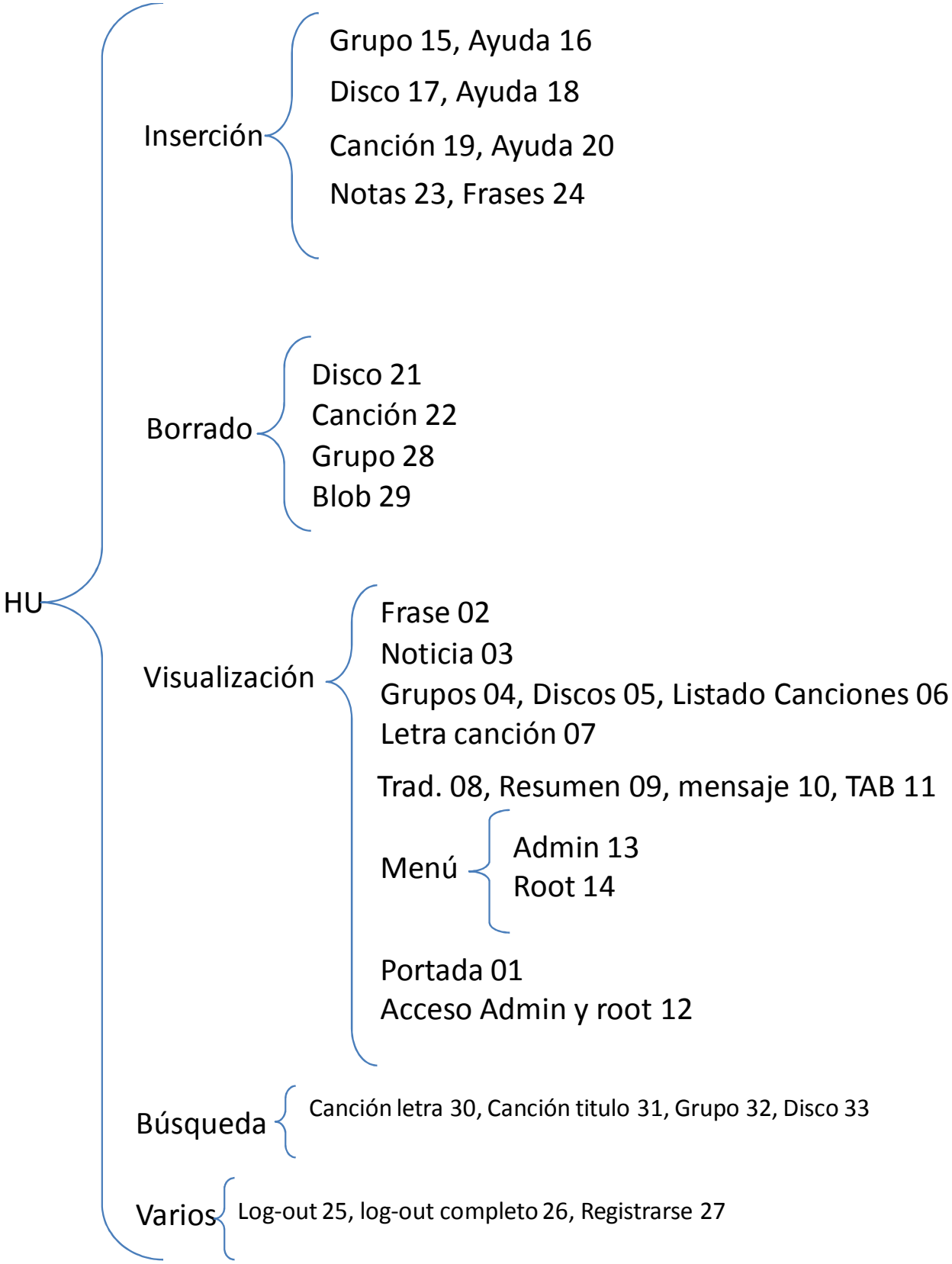


Ilustración 3-35 Resumen Historias de Usuario

HU-01			
Título	Portada		
Prioridad	100	Estimación	8
Enunciado	Presentación de la aplicación		
Dependencias	-		
Criterios de Aceptación	▪ Se ven los grupos juntos con sus datos y se puede navegar por ellos. ▪ Se ven noticias. ▪ Se ven frases. ▪ Acceso al menú principal.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación del código cliente. ▪ Implementación del código servidor. ▪ Pruebas de Aceptación asociadas.		
Pruebas de Aceptación	PA-12, PA-02, PA-03		

Tabla 3-9 HU Portada

HU-02			
Título	Ver todas las frases célebres		
Prioridad	50	Estimación	5
Enunciado	Ver todas las frases célebres y navegar por ellas.		
Dependencias	HU-24		
Criterios de Aceptación	▪ Se ven las frases en carrusel automáticamente o al hacer clic en siguiente o en anterior. Enlaces externos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-21		

Tabla 3-10 HU Ver todas las Frases Célebres

HU-03			
Título	Ver todas las noticias		
Prioridad	50	Estimación	5
Enunciado	Ver todas las noticas y navegar por ellas.		
Dependencias	HU-23		
Criterios de Aceptación	▪ Se ven las noticas en carrusel automáticamente o al hacer clic en siguiente o en anterior. Enlaces externos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-20		

Tabla 3-11 HU Ver todas las Noticias

HU-04			
Título	Ver los datos de los grupos		
Prioridad	100	Estimación	5
Enunciado	Ver los datos de los grupos y navegar por ellos y sus datos.		
Dependencias	HU-15		
Criterios de Aceptación	▪ Se ven los grupos en carrusel automáticamente o al hacer clic en siguiente o en anterior. Enlaces externos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-12		

Tabla 3-12 HU Ver los Datos de los Grupos

HU-05			
Título	Ver los discos de un grupo		
Prioridad	100	Estimación	5
Enunciado	Ver los discos y navegar por ellos y sus datos.		
Dependencias	HU-17		
Criterios de Aceptación	▪ Se ven los discos en carrusel automáticamente o al hacer clic en siguiente o en anterior. Enlaces externos. Se puede volver al grupo del cual son los discos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-13		

Tabla 3-13 HU Ver los Discos de un Grupo

HU-06			
Título	Ver las canciones de un disco de grupo		
Prioridad	100	Estimación	5
Enunciado	Ver la lista de canciones de un disco de un grupo y sus datos.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve la portada de un disco y un listado de las canciones del mismo, se puede volver al grupo.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-14		

Tabla 3-14 HU Ver las Canciones de un Disco de Grupo

HU-07			
Título	Ver los datos de una canción de un disco de grupo (Letra)		
Prioridad	100	Estimación	5
Enunciado	Ver datos de canciones de un disco de un grupo.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve la información de una canción de un determinado disco de un grupo dado y se puede volver al disco del que forma parte.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-15		

Tabla 3-15 HU Ver los Datos de una Canción de un Disco de Grupo (Letra)

HU-08			
Título	Ver la letra traducida de una canción de un disco de grupo		
Prioridad	100	Estimación	1
Enunciado	Ver la letra de una canción de un disco de un grupo.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve la letra de una canción de un determinado disco de un grupo dado.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-16		

Tabla 3-16 HU Ver la Letra traducida de una Canción de un Disco de Grupo

HU-09			
Título	Ver un resumen de la letra de una canción de un disco de grupo		
Prioridad	80	Estimación	1
Enunciado	Ver resumen de la letra traducida de una canción de un disco de un grupo.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve un resumen traducido de una canción de un determinado disco de un grupo dado.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación		
Pruebas de Aceptación	PA-17		

Tabla 3-17 HU Ver un Resumen de la Letra de una Canción de un Disco de Grupo

HU-10			
Título	Ver mensaje de una canción de un disco de grupo		
Prioridad	80	Estimación	1
Enunciado	Ver el mensaje traducido de una canción de un disco de un grupo.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve el mensaje traducido de una canción de un determinado disco de un grupo dado.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-18		

Tabla 3-18 HU Ver Mensaje de una Canción de un Disco de Grupo

HU-11			
Título	Ver el TAB de una canción de un disco de grupo		
Prioridad	80	Estimación	1
Enunciado	Ver el TAB de una canción de un disco de un grupo.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Se ve el TAB de una canción de un determinado disco de un grupo dado.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-19		

Tabla 3-19 HU Ver el TAB de una Canción de un Disco de Grupo

HU-12			
Título	Acceso como Administrador		
Prioridad	100	Estimación	5
Enunciado	Poder acceder a las operaciones de un administrador.		
Dependencias	HU-01		
Criterios de Aceptación	▪ Autenticación en el sistema como administrador.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-22		

Tabla 3-20 HU Acceso como Administrador

HU-13			
Título	Menú propio del Administrador		
Prioridad	100	Estimación	1
Enunciado	Poder realizar las operaciones de un administrador.		
Dependencias	-		
Criterios de Aceptación	▪ Ver sólo opciones permitidas a los administradores.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-23		

Tabla 3-21 HU Menú propio del Administrador

HU-14			
Título	Menú como Root (Puede insertar Grupo)		
Prioridad	100	Estimación	1
Enunciado	Poder realizar las operaciones propias de Root.		
Dependencias	-		
Criterios de Aceptación	▪ Autenticación en el sistema como Root.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-24		

Tabla 3-22 HU Menú como Root (Puede insertar Grupo)

HU-15			
Título	Insertar un grupo		
Prioridad	100	Estimación	8
Enunciado	Inserta un grupo, con todos su datos en el sistema.		
Dependencias	-		
Criterios de Aceptación	▪ Sólo Root puede. ▪ No insertar grupo repetido. ▪ Ni grupo no autorizado.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-01		

Tabla 3-23 Historia de Usuario: Insertar un Grupo

HU-16			
Título	Ayuda y vídeos... a insertar un grupo		
Prioridad	25	Estimación	5
Enunciado	Ayuda a insertar grupo.		
Dependencias	-		
Criterios de Aceptación	▪ Ayuda.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-02		

Tabla 3-24 HU Ayuda y vídeos... a insertar un grupo

HU-17			
Título	Insertar un disco		
Prioridad	100	Estimación	8
Enunciado	Inserta un disco con todos sus datos.		
Dependencias	HU-15		
Criterios de Aceptación	▪ Sólo si es administrador de ese grupo. No repetidos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-03		

Tabla 3-25 HU Insertar un Disco

UH-18			
Título	Ayuda y vídeos... a insertar un disco		
Prioridad	25	Estimación	1
Enunciado	Ayuda a insertar disco.		
Dependencias	-		
Criterios de Aceptación	▪ Ayuda.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-04		

Tabla 3-26 HU Ayuda y vídeos... a insertar un Disco

HU-19			
Título	Insertar una canción		
Prioridad	100	Estimación	8
Enunciado	Inserta una canción a un disco de un grupo.		
Dependencias	HU-17		
Criterios de Aceptación	▪ Si es administrador de ese grupo. No repetidos.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-05		

Tabla 3-27 HU Insertar una Canción

HU-20			
Título	Ayuda y videos... a insertar una canción		
Prioridad	25	Estimación	1
Enunciado	Ayuda a insertar una canción.		
Dependencias	-		
Criterios de Aceptación	▪ Ayuda.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-06		

Tabla 3-28 HU Ayuda y vídeos... a insertar una Canción

HU-21			
Título	Borrar disco		
Prioridad	25	Estimación	5
Enunciado	Borra un disco de un grupo asignado al administrador.		
Dependencias	HU-17		
Criterios de Aceptación	▪ Sólo el administrador de ese grupo puede hacerlo.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-09		

Tabla 3-29 HU Borrar Disco

HU-22			
Título	Borra una canción		
Prioridad	25	Estimación	5
Enunciado	Borra una canción de un grupo de un disco.		
Dependencias	HU-19		
Criterios de Aceptación	▪ Sólo si es el administrador de ese grupo.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-10		

Tabla 3-30 HU Borra una Canción

HU-23			
Título	Insertar noticia		
Prioridad	75	Estimación	5
Enunciado	Inserta una noticia en el sistema.		
Dependencias	-		
Criterios de Aceptación	▪ Sólo los administradores pueden hacerlo.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	AT-07		

Tabla 3-31 HU Insertar Noticia

HU-24			
Título	Insertar una frase		
Prioridad	25	Estimación	5
Enunciado	Inserta frase.		
Dependencias	-		
Criterios de Aceptación	▪ Sólo los administradores pueden hacerlo.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-08		

Tabla 3-32 HU Insertar una Frase

HU-25			
Título	Cierre de sesión (Log-out)		
Prioridad	100	Estimación	5
Enunciado	Cierra la sesión con el usuario actual.		
Dependencias	HU-12		
Criterios de Aceptación	▪ Se Desconecta del usuario actual.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-30		

Tabla 3-33 HU Cierre de Sesión (Log-out)

HU-26			
Título	Cierre de sesión completo		
Prioridad	100	Estimación	1
Enunciado	Cierra la sesión con el navegador con el usuario actual.		
Dependencias	-		
Criterios de Aceptación	▪ Se desconecta totalmente del usuario actual.		
Tareas	▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-31		

Tabla 3-34 HU Cierre de Sesión completo

HU-27			
Título	Registrarse en algún grupo/s		
Prioridad	100	Estimación	5
Enunciado	Ser informado de los cambios en algún grupo.		
Dependencias	-		
Criterios de Aceptación	▪ Autenticación en el sistema como administrador.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-25		

Tabla 3-35 HU Registrarse en algún Grupo/s

HU-28			
Título	Borrar un grupo		
Prioridad	50	Estimación	1
Enunciado	Borrar un grupo por parte de SuperRoot.		
Dependencias	HU-15		
Criterios de Aceptación	▪ Elimina el grupo del sistema desde consola.		
Tareas	▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-11		

Tabla 3-36 HU Borrar un Grupo

HU-29			
Título	Borrar logos o portadas		
Prioridad	25	Estimación	5
Enunciado	Borra un logo o una portada de un disco.		
Dependencias	HU-15, HU-17		
Criterios de Aceptación	▪ Elimina los datos.		
Tareas	▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-10,PA-11		

Tabla 3-37 HU Borrar Logos o Portadas

HU-30			
Título	Búsqueda de canciones		
Prioridad	100	Estimación	8
Enunciado	Busca todas las canciones que contengan en la letra el texto indicado.		
Dependencias	HU 15, HU 17, HU 19		
Criterios de Aceptación	▪ Aparece el listado o indica que no hay.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-26		

Tabla 3-38 HU Búsqueda de Canciones

HU-31			
Título	Búsqueda de canciones por título		
Prioridad	100	Estimación	8
Enunciado	Busca todas las canciones que contengan en el título el texto indicado.		
Dependencias	HU 15, HU 17, HU 19		
Criterios de Aceptación	▪ Aparece el listado o indica que no hay.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-27		

Tabla 3-39 HU Búsqueda de Canciones por Título

HU-32			
Título	Búsqueda de grupos		
Prioridad	100	Estimación	5
Enunciado	Busca todas las grupos que contengan en el nombre el texto indicado.		
Dependencias	HU-15		
Criterios de Aceptación	▪ Aparece el carrusel de todos los grupos o dice que no hay.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-28		

Tabla 3-40 HU Búsqueda de Grupos

HU-33			
Título	Búsqueda de discos		
Prioridad	100	Estimación	5
Enunciado	Busca todas los discos que contengan en el título el texto indicado.		
Dependencias	HU-17		
Criterios de Aceptación	▪ Aparece el listado o indica que no hay.		
Tareas	▪ Diseño de la Interfaz. ▪ Implementación de código cliente. ▪ Implementación de código servidor. ▪ Pruebas de Aceptación.		
Pruebas de Aceptación	PA-29		

Tabla 3-41 HU Búsqueda de Discos

3.5 BOOTSTRAP

Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT en el año 2011 y su desarrollo continua en un repositorio de GitHub.

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

Desde la aparición de Bootstrap 3, el framework se ha vuelto bastante más compatible con desarrollo web responsive, entre otras características se han reforzado las siguientes:

- Soporte bastante bueno (casi completo) con HTML5 y CSS3, permitiendo ser usado de forma muy flexible para desarrollo web con unos excelentes resultados.
- Se ha añadido un sistema GRID que permite diseñar usando un GRID de 12 columnas, donde se debe plasmar el contenido, con esto podemos desarrollar responsive de forma mucho más fácil e intuitiva.
- Boostrap 3 establece Media Queries para 4 tamaños de dispositivos diferentes, variando dependiendo del tamaño de su pantalla, estas Media Queries permiten desarrollar para dispositivos móviles y tablets de forma mucho más fácil.
- Boostrap 3 también permite insertar imágenes responsive, es decir, con sólo insertar la imagen con la clase “img-responsive” las imágenes se adaptaran al tamaño.

Todas estas características hacen que Boostrap sea una excelente opción para desarrollar webs y aplicaciones web totalmente adaptables a cualquier tipo de dispositivo.

Boostrap es compatible con la mayoría de navegadores web del mercado, y más desde la versión 3, actualmente es totalmente compatible con los siguientes navegadores:

- Google Chrome (en todas las plataformas).
- Safari (tanto en iOS como en Mac).
- Mozilla Firefox (en Mac y en Windows).
- Internet Explorer (en Windows y Windows Phone).
- Opera (en Windows y Mac).

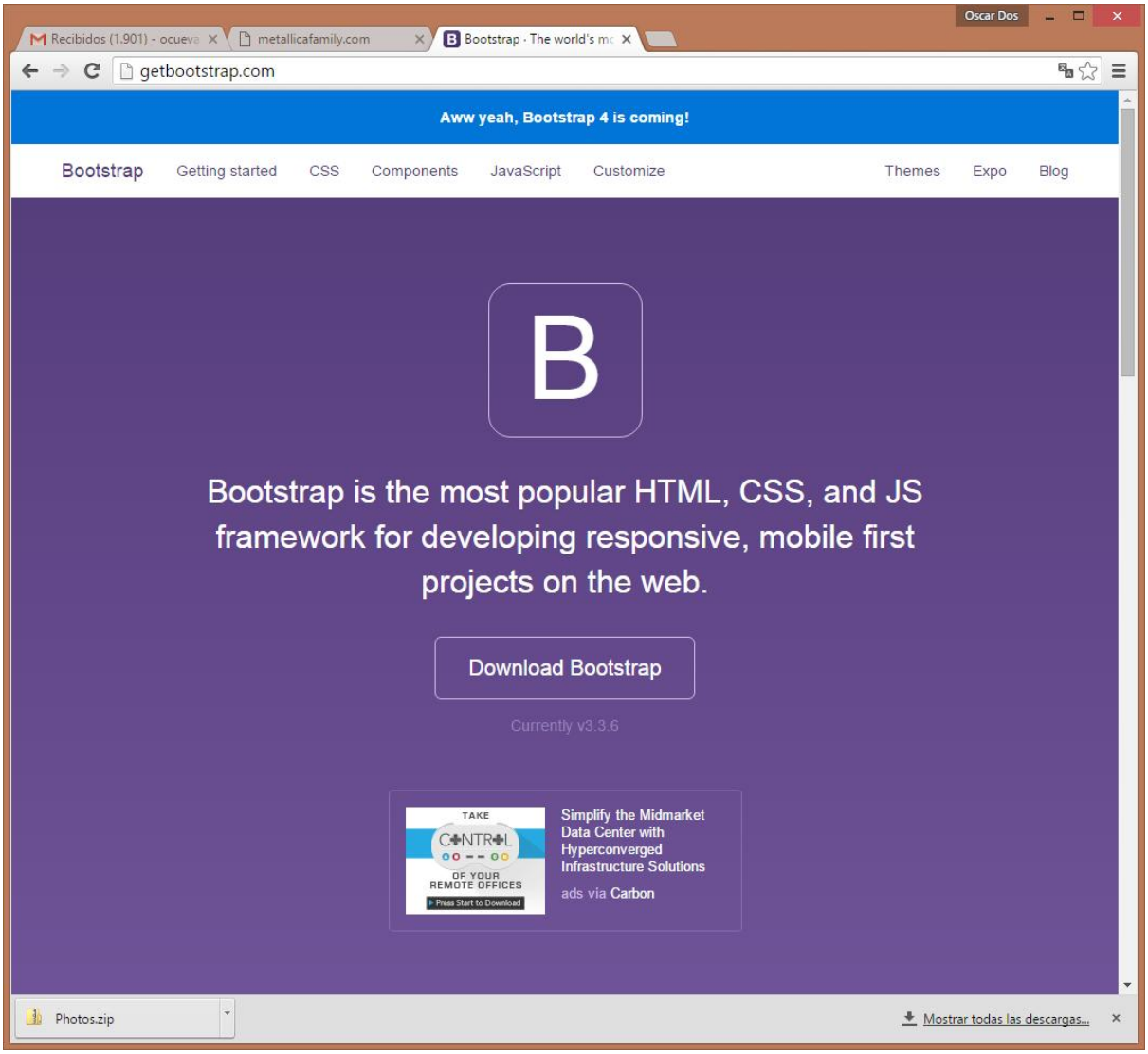


Ilustración 3-36 Bootstrap

Una web interesante para crear interfaces bootstrap es layoutit que permite, simplemente usando arrastrar y soltar (Drag-and-drop), utilizar los componentes de **Bootstrap** en el diseño de la interfaz.

Es fácil de integrar con **cualquier lenguaje de programación**, solamente hay que bajar el HTML generado y comenzar a programar el diseño, genera código HTML profesional y validado. Los diseños son **CSS Responsive** y **Fluid.**

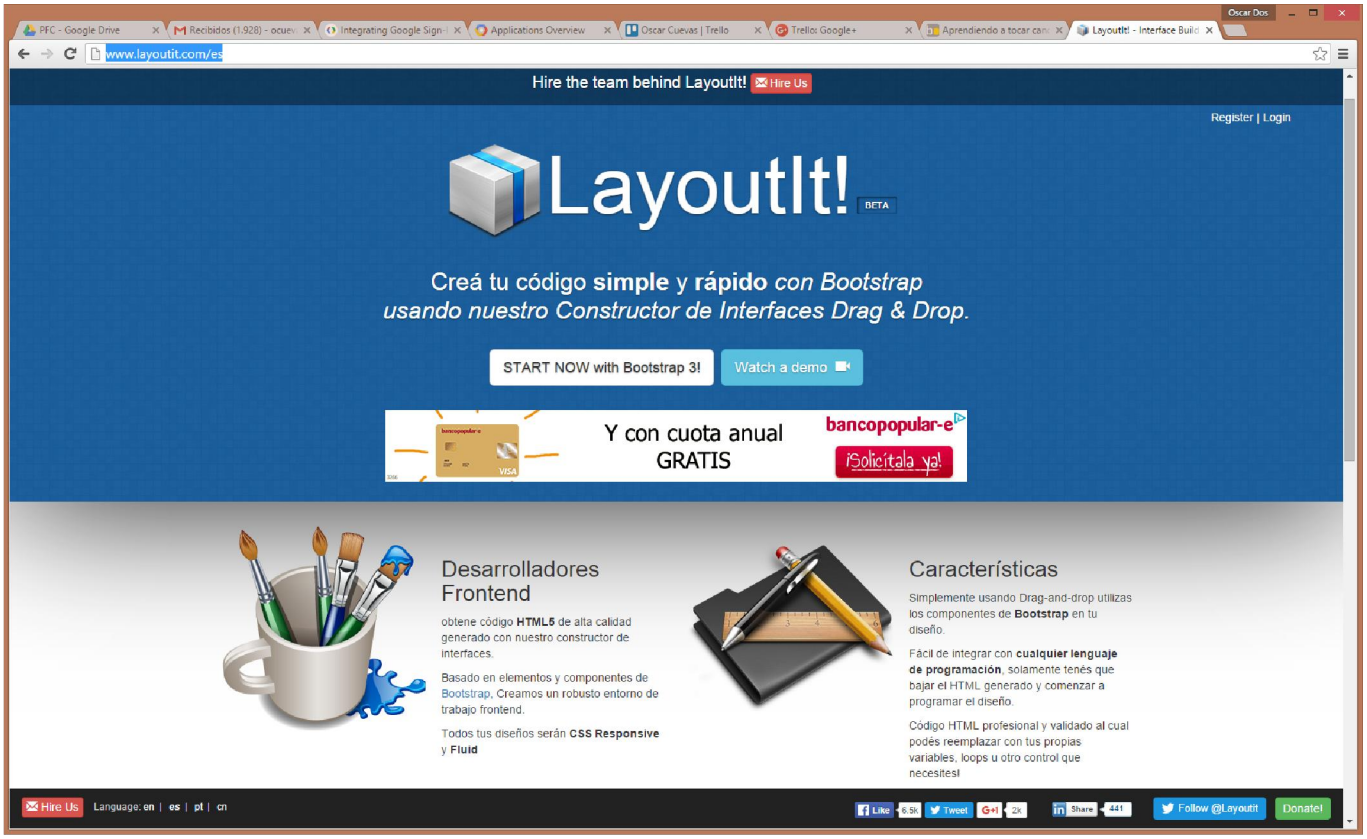


Ilustración 3-37 http://www.layoutit.com/es

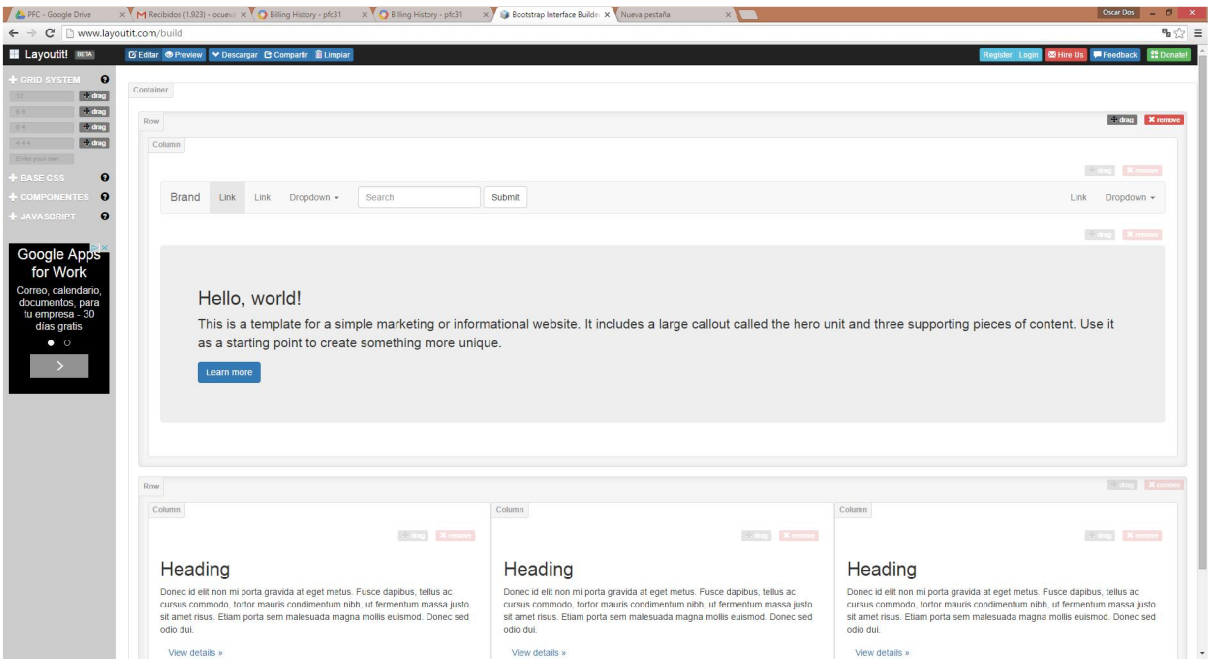


Ilustración 3-38 Ejemplo de uso de Layoutit

3.6 METODOLOGÍA

En el apartado 2.2 Metodología se exponían los tipos de metodologías de desarrollo de software. Las metodologías tradicionales y las ágiles, comentando las ventajas e inconvenientes de cada una.
Finalmente se ha escogido una metodología ágil para la planificación del proyecto, en concreto se ha utilizado Scrum. Esta decisión se ha tomado en base a que el proyecto a desarrollar no es muy largo, permite unas fechas flexibles y la disponibilidad del Product Owner. Otro de los motivos para la elección de esta metodología ha sido el conocimiento de Scrum y la popularidad actual con la que cuenta.

3.6.1 SCRUM, LA METODOLOGÍA ÁGIL MÁS EXTENDIDA

De las diversas metodologías ágiles existentes en el mercado, se hará uso de Scrum, ya que según el informe anual que publica VersionOne, esta metodología es la más extendida en las organizaciones según el voto de 4.048 personas de la industria del software.

Scrum se basa en tres reglas fundamentales:

- **Transparencia**: todos los aspectos del proceso que afectan al resultado del proyecto son visibles para todas aquellas personas que administran el resultado. Un ejemplo de ello son las técnicas colaborativas que se usan para la comunicación entre los diferentes miembros del equipo.

- **Inspección**: se debe controlar continuamente todos los aspectos del proceso de Scrum. Una manera de controlarlos es mediante las reuniones del equipo.

- **Adaptación**: en caso de desviación de los objetivos, se procederá a una adaptación del proceso para llegar al objetivo final.

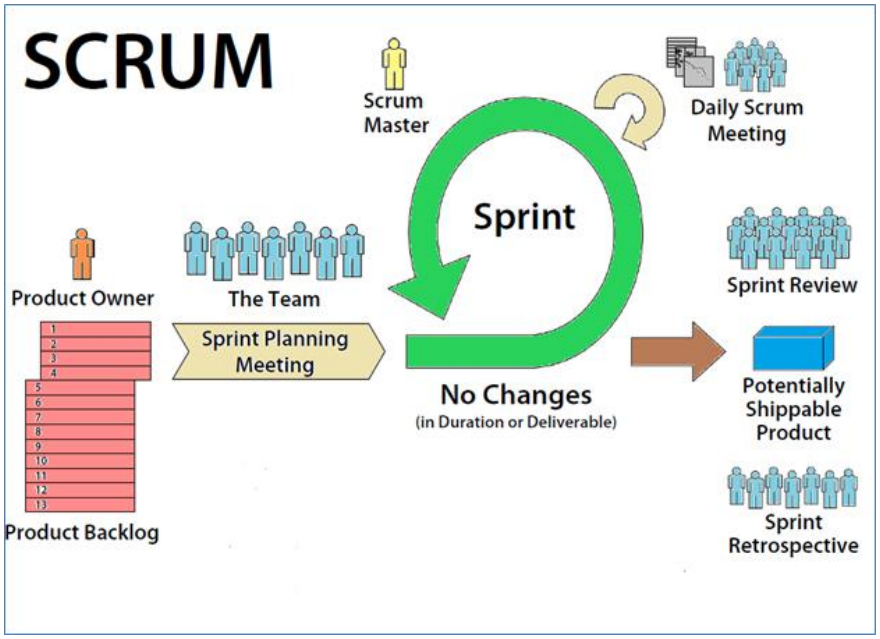


Ilustración 3-39 Ciclo de vida Scrum

3.6.2 EQUIPO DE TRABAJO

En la metodología ágil Scrum existen tres roles que son: el Product Owner, persona que conoce lo que hay que desarrollar y el orden de desarrollo; el Scrum Master, que se encarga de hacer que las reglas se cumplan; y el equipo de desarrollo, encargado de desarrollar el sistema.

3.6.3 El PRODUCT OWNER

Este rol es el responsable de gestionar las necesidades que serán satisfechas por el proyecto y asegurar el valor del trabajo que el equipo lleva a cabo. Las principales responsabilidades son las siguientes:

- Definición de las historias de usuario, priorización y fijar criterios de aceptación de las mismas.
- Ordenar los elementos del Product Backlog.
- Acordar junto con el resto del equipo la definición de “cuando algo está terminado”. En Scrum viene denominado por “DONE”.

3.6.4 El SCRUM MASTER

El Scrum Master es el responsable de asegurarse de que todo el equipo entiende Scrum y actúa en consecuencia siguiendo sus prácticas y reglas.

El apoyo que presta el Scrum Master al Product Owner es el siguiente:

- Le enseña técnicas para gestionar el Product Backlog.
- Enseña al Product Owner el concepto de agilidad.

El Scrum Master ayuda al equipo de desarrollo de la siguiente manera:

- Entender los elementos del Product Backlog.
- Crear productos que aporten valor al cliente.
- Eliminar los problemas que impiden el progreso.

3.6.5 EQUIPO DE DESARROLLO

Está formado por los diferentes desarrolladores que convierten las necesidades del Product Owner en un conjunto de funcionalidades del producto software final. Para que el equipo de desarrollo funcione de la manera correcta deberá auto-gestionarse de tal manera que:

- Sea autónomo: la toma de decisiones se hace por el equipo en conjunto.
- Sea adaptable: el equipo se ajusta de la mejor manera para resolver los problemas.
- Sea responsable: todos los miembros del equipo comparten las responsabilidades de los resultados.

3.6.6 EQUIPO DE TRABAJO DEL PROYECTO

El equipo de trabajo del proyecto está compuesto por dos personas:

- Jesús Hernando Corrochano, tutor del proyecto, que actúa como Product Owner y Scrum Master.
- Óscar Cuevas Lanchares, que actúa como equipo de desarrollo auto-gestión y como Product Owner.

3.6.7 EVENTOS EN SCRUM

Los eventos en Scrum tienen como objetivo crear regularidad y minimizar la necesidad de reuniones no definidas en la metodología.

Los eventos son los siguientes:

Sprint: bloque de tiempo de un mes o menos durante el cual se crea un incremento del producto que pasa a ser utilizable y potencialmente desplegable. Cada Sprint debe comenzar inmediatamente después de la finalización del Sprint previo. La duración de cada Sprint debe ser prefijada antes del comienzo del proyecto. Los Sprints están compuestos por Reunión de Sprint Planning Meeting "Planificación del Sprint", Daily Scrums "Scrums Diarios", el trabajo de desarrollo, Sprint Review "la Revisión del Sprint" y el Sprint Retrospective "Retrospectiva del Sprint".

En el desarrollo del proyecto realizado, se ha planificado un total de cuatro Sprints de una duración media de un mes (sin contar días festivos ni fines de semana).

Sprint Planning Meeting: reunión del equipo para planificar el trabajo a realizar en el Sprint. Con una duración máxima de ocho horas para un Sprint de un mes, se responden a dos preguntas:

1. ¿Qué puede ser terminado para el Sprint?
2. ¿Cómo se conseguirá completar el trabajo a realizar?

A lo largo del proyecto, se han realizado un total de cuatro reuniones presenciales tras finalizar cada Sprint entre el Product Owner / Scrum Master (Jesús Hernando Corrochano) y el equipo de desarrollo (Óscar Cuevas Lanchares).

Daily Scrum: reunión diaria de 15 minutos sobre el estado del proyecto para que el equipo de desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas. Durante la reunión cada miembro del equipo de desarrollo tiene que contestar a tres preguntas.

1. ¿Qué has realizado desde ayer?
2. ¿Qué es lo que haré hasta la reunión de mañana?
3. ¿He tenido algún problema que me ha impedido alcanzar el objetivo?

Para el desarrollo del proyecto, se han llevado a cabo reuniones diarias (de Lunes a Viernes) a través de correo y whatsapp entre el equipo del proyecto.

Sprint Review: Reunión informal del equipo completo junto con los interesados para inspeccionar lo realizado a lo largo del Sprint. Tiene una duración máxima de cuatro horas para un Sprint de un mes y el objetivo es:

1. Presentar el trabajo completado a los interesados.
2. Revisar el trabajo que fue completado y el no completado.

A lo largo del proyecto desarrollado, se han realizado un total de cuatro reuniones presenciales entre el equipo de trabajo, una vez se finalizó cada Sprint.

Sprint Retrospective: La retrospectiva del Sprint tiene como objetivo realizar una reunión de los miembros del equipo para que compartan sus impresiones sobre el Sprint que acaban de terminar, de tal manera que se pueda crear un plan de mejoras para el siguiente Sprint. La reunión tiene un tiempo de tres horas para los Sprints de un mes.

A lo largo del proyecto desarrollado, se han realizado un total de cuatro reuniones presenciales entre el equipo de trabajo, una vez se finalizó cada Sprint.

3.6.8 ARTEFACTOS EN SCRUM

Los artefactos de Scrum sirven para representar trabajo de tal forma que el equipo completo de Scrum tenga una visión transparente del proyecto.

Product Backlog: Es un listado ordenado y priorizado de todo lo que se quiere añadir al producto, donde, generalmente las historias de usuario son el elemento más común en la lista. El responsable del Product Backlog es el Product Owner, el cual se encarga de alimentar, ordenar y mantener el listado.

El descubrimiento de nuevos elementos del Product Backlog es un proceso continuo y, a diferencia del ciclo de vida en cascada, está en continuo crecimiento a lo largo de todo el proyecto Scrum.

En el proyecto desarrollado, el Product Backlog se ha creado mediante un listado de las diferentes historias de usuario que se han ido añadiendo al proyecto. Se ha hecho uso de una herramienta colaborativa denominada Trello[xlvi].

Ilustración 3-40 http://trello.com

Sprint Backlog: Constituye una lista de elementos del Product Backlog que han sido seleccionados para desarrollar en el Sprint actual. A medida que el trabajo se completa se va actualizando la lista de elementos. Para el listado de elementos que se ha ido desarrollando a lo largo de cada Sprint, también se ha hecho uso de la herramienta Trello.

Ciclo de vida iterativo e incremental: Corresponde con suma de todos los elementos del Product Backlog completados durante un Sprint y los Sprints anteriores.

Product Backlog: listado ordenado de historias de usuario aún no implementadas ni pertenecientes a ningún Sprint.

Grooming: listado historias de usuario que se podrán realizar en el siguiente Sprint.

- In Sprint: listado de historias de usuario del Sprint actual.
- Done: listado de historias de usuario completadas.
- Bibliography: bibliografía que se ha hecho referencia.

Para la gestión del ciclo de vida Scrum se ha utilizado la herramienta Trello. Una de las razones por las que se decidió utilizar esta herramienta, además de por su sencillez, fue la capacidad de añadir tableros y tareas por parte del tutor vía email. De esta forma se facilitaba la comunicación entre ambas partes.

El tablero del proyecto se ha dividido en seis columnas:

- **Pendiente**: Historias de Usuario pendientes de realizar fuera del Sprint Actual. Estarán agrupadas en colores según el Sprint Backlog, dicho color puede variar durante el ciclo de vida del proyecto.
- **Grooming**: Historias de Usuario que serán realizadas en el siguiente Sprint.
- **Sprint Actual**: Historias de usuario del Sprint Actual que aún no han sido empezadas.
- **En Desarrollo**: Historias de usuario del Sprint Actual que están siendo elaboradas.
- **Pruebas**: Historias de Usuario que ya han sido desarrolladas, pero que deben pasar las pruebas necesarias para garantizar que cumplen los criterios de aceptación.
- **Finalizado**: Historias finalizadas independientemente del Sprint al que pertenezcan.

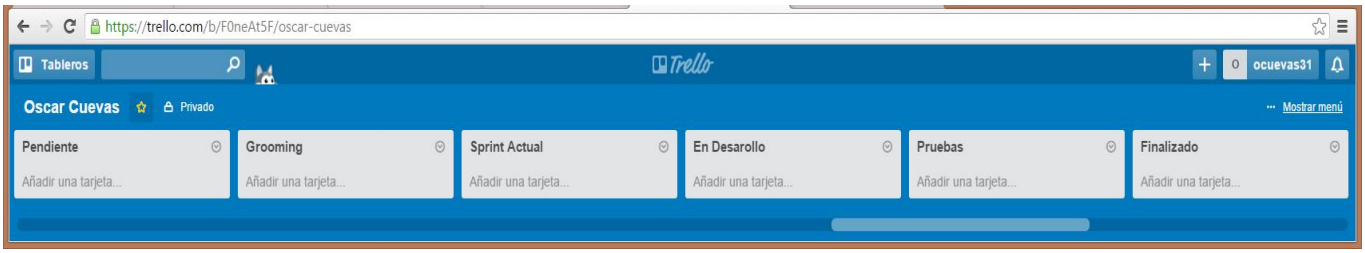


Ilustración 3-41 Tablero del Proyecto

3.7 PRUEBAS

Se han automatizado en la medida de lo posible las pruebas, utilizando la herramienta Selenium[xlvii], que permite realizar una serie de operaciones sobre una web y grabarlas para poder repetirlas las veces necesarias. Por ejemplo, para insertar un grupo y comprobar que se realizaba de manera correcta, se grababa la inserción y si no funcionaba no hacía falta volver a rellenar todos los datos, sino que se ejecutaba la grabación realizada en Selenium.

Pruebas con distintos dispositivos, ordenadores personales con diferentes resoluciones, portátiles, tablets, móviles. También con distintos Sistemas Operativos (Linux, Windows, Android) y distintos navegadores (Explorer, Chrome, Opera, Firefox).

Selenium para pruebas:

Ilustración 3-42 http://www.seleniumhq.org/

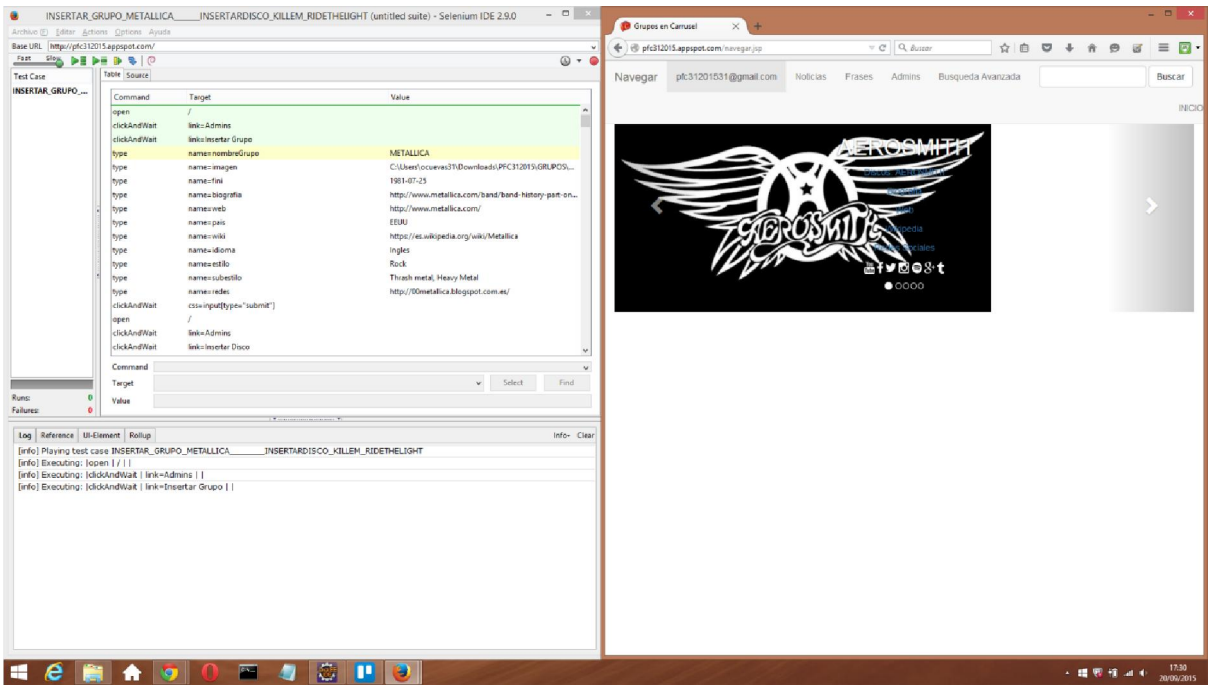


Ilustración 3-43 Ejemplo de uso de Selenium para Insertar un Grupo

A continuación se describen las pruebas de Aceptación realizadas para verificar el cumplimiento de las historias de usuario del proyecto.

Cada prueba de aceptación contendrá la siguiente información:

- **Identificador**: Identificador unívoco de la prueba. Tiene el formato PA-XX, donde 'PA' corresponde al acrónimo de Pruebas de Aceptación (Acceptance Test) y 'XX' corresponde al número de la prueba.
- **Nombre**: Nombre característico de cada prueba.
- **Descripción**: Explicación de la prueba.
- **Procedimiento**: Pasos realizados para realizar la prueba.
- **Historia de Usuario**: Historia de usuario que verifica la prueba de aceptación.
- **Resultado Esperado**: Describe el resultado que se espera al finalizar la prueba.
- **Estado**: Resultado de la prueba.

PA-01	
Nombre	Inserción de un Grupo
Descripción	Se comprueba que se puede insertar toda la información de un grupo en el sistema.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre insertar Grupo. ▪ El usuario rellena correctamente el formulario de inserción de un grupo.
Historia de Usuario	HU-15
Resultado Esperado	Si el usuario no es root no se permite. Si el grupo ya ha sido insertado no se permite. Si el grupo no está autorizado a insertarse no se permite. Si faltan datos imprescindibles del grupo no se permite. Si se permite, envía correo a root.
Estado	Éxito. El grupo se inserta correctamente y se envía correo a root.

Tabla 3-42 PA Inserción de un Grupo

PA-02	
Nombre	Ayuda a la Inserción de un Grupo
Descripción	Se comprueba que se puede ver la ayuda para insertar correctamente un grupo en el sistema.
Procedimiento	▪ El usuario pulsa sobre Ayuda.
Historia de Usuario	HU-16
Resultado Esperado	Ayuda a la inserción de un grupo.
Estado	Éxito. Se ve la ayuda para insertar un grupo.

Tabla 3-43 PA Ayuda a la Inserción de un Grupo

PA-03	
Nombre	Inserción de un disco
Descripción	Se comprueba que se puede insertar toda la información de un disco de un grupo en el sistema.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre Insertar Disco. ▪ El usuario rellena correctamente el formulario de inserción de un disco de un grupo.
Historia de Usuario	HU-17
Resultado Esperado	Si el usuario no es administrador de ese grupo no se permite. Sólo aparecen los grupos asignados al administrador en cuestión. Si el disco ya ha sido insertado no se permite. Si faltan datos imprescindibles del disco no se permite. Si se permite, envía correo a root y a todos los usuarios registrados a ese grupo.
Estado	Éxito. El grupo se inserta correctamente y se envía correo a root y a usuarios registrados a ese grupo.

Tabla 3-44 PA Inserción de un Disco

PA-04	
Nombre	Ayuda a la inserción de un disco
Descripción	Se comprueba que se puede ver la ayuda para insertar correctamente un disco en el sistema.
Procedimiento	▪ El usuario pulsa sobre Ayuda.
Historia de Usuario	HU-18
Resultado Esperado	Ayuda a la inserción de un disco.
Estado	Éxito. Se ve la ayuda para insertar un disco.

Tabla 3-45 PA Ayuda a la Inserción de un Disco

PA-05	
Nombre	Inserción de una canción de un disco de un Grupo
Descripción	Se comprueba que se puede insertar toda la información de un grupo en el sistema incluida la letra y el TAB largos.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre insertar canción. ▪ El usuario rellena correctamente el formulario de inserción de una canción.
Historia de Usuario	HU-19
Resultado Esperado	Si el usuario no es administrador de ese grupo no se permite, sólo aparecen los grupos del administrador en cuestión y sólo los discos de ese grupo. Si la canción ya ha sido insertada no se permite. Si faltan datos imprescindibles de la canción no se permite. Se pueden insertar letras muy largas y TABS también largos. Si se permite, envía correo a root y a todos los usuarios registrados a ese grupo.
Estado	Éxito. La canción se inserta correctamente y se envía correo a root y a todos los usuarios registrados a ese grupo.

Tabla 3-46 PA Inserción de una Canción de un Disco de un Grupo

PA-06	
Nombre	Ayuda a la Inserción de una Canción
Descripción	Se comprueba que se puede ver la ayuda para insertar correctamente una canción en el sistema
Procedimiento	▪ El usuario pulsa sobre Ayuda.
Historia de Usuario	HU-20
Resultado Esperado	Ayuda a la inserción de una canción.
Estado	Éxito. Se ve la ayuda para insertar una canción.

Tabla 3-47 PA Ayuda a la Inserción de una Canción

PA-07	
Nombre	Inserción de una noticia
Descripción	Se comprueba que se puede insertar una noticia con enlaces en el sistema.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre insertar noticia. ▪ El usuario rellena correctamente el formulario de inserción de una noticia.
Historia de Usuario	HU-23
Resultado Esperado	Si el usuario no es administrador no se permite. Si se permite, envía correo a root.
Estado	Éxito. La noticia con enlaces se inserta correctamente y se envía correo a root.

Tabla 3-48 PA Inserción de una Noticia

PA-08	
Nombre	Inserción de una frase
Descripción	Se comprueba que se puede insertar una frase con enlaces en el sistema.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre insertar frase. ▪ El usuario rellena correctamente el formulario de inserción de una frase.
Historia de Usuario	HU-24
Resultado Esperado	Si el usuario no es administrador no se permite. Si se permite, envía correo a root.
Estado	Éxito. La frase con enlaces se inserta correctamente y se envía correo a root.

Tabla 3-49 PA Inserción de una Frase

PA-09	
Nombre	Borrar un disco de un grupo
Descripción	Se comprueba que se puede eliminar un disco de un grupo.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre borrar disco. ▪ El usuario selecciona el grupo sólo entre los que tiene asignados y selecciona el disco dentro de los discos del grupo seleccionado con anterioridad.
Historia de Usuario	HU-21
Resultado Esperado	Si el usuario no es administrador no se permite. Si se permite, envía correo a root.
Estado	Éxito. Si se elimina correctamente y se envía correo a root.

Tabla 3-50 PA Borrar un Disco de un Grupo

PA-10	
Nombre	Borrar una canción de un disco de un grupo
Descripción	Se comprueba que se puede eliminar una canción de un disco de un grupo.
Procedimiento	▪ El usuario pulsa sobre Admin y sobre borrar canción. ▪ El usuario selecciona el grupo sólo entre los que tiene asignados y selecciona el disco dentro de los discos del grupo seleccionado con anterioridad y selecciona la canción a borrar dentro de las canciones existentes de ese disco de ese grupo.
Historia de Usuario	HU-22
Resultado Esperado	Si el usuario no es administrador no se permite. Si se permite, envía correo a root.
Estado	Éxito. Si se elimina correctamente y se envía correo a root.

Tabla 3-51 PA Borrar una Canción de un Disco de un Grupo

PA-11	
Nombre	Borrar un grupo
Descripción	Se comprueba que se puede eliminar un grupo desde consola.
Procedimiento	El súper-usuario accede a consola en https://appengine.google.com/deployment?&app_id=e~pfc312015 y pulsa sobre Datastore Viewer, selecciona Grupo, marca el grupo a borrar y pulsa sobre Delete.
Historia de Usuario	HU-28
Resultado Esperado	Si el usuario no es súper-usuario no se permite. Se deben eliminar antes discos y canciones.
Estado	Éxito. Si se elimina correctamente.

Tabla 3-52 PA Borrar un Grupo

PA-12	
Nombre	Visualización de los Grupos
Descripción	Se comprueba que se puede visualizar toda la información de los grupos y los enlaces externos.
Procedimiento	▪ El usuario pulsa sobre Navegar o busca grupos.
Historia de Usuario	HU-04
Resultado Esperado	Ver los datos de los grupos.
Estado	Éxito.

Tabla 3-53 PA Visualización de los Grupos

PA-13	
Nombre	Visualización de los Discos
Descripción	Se comprueba que se puede visualizar toda la información de los discos y los enlaces externos.
Procedimiento	▪ El usuario pulsa sobre discos de tal grupo o busca discos.
Historia de Usuario	HU-05
Resultado Esperado	Ver los datos de los discos.
Estado	Éxito.

Tabla 3-54 PA Visualización de los Discos

PA-14	
Nombre	Visualización de las canciones
Descripción	Se comprueba que se puede visualizar toda la información de las canciones de un disco y los enlaces externos.
Procedimiento	▪ El usuario pulsa sobre canciones de tal disco o busca canciones.
Historia de Usuario	HU-06, HU-07
Resultado Esperado	Ver los datos de las canciones.
Estado	Éxito.

Tabla 3-55 PA Visualización de las Canciones

PA-15	
Nombre	Visualización de toda la letra de una canción
Descripción	Se comprueba que se puede visualizar la letra original de la canción.
Procedimiento	▪ El usuario pulsa sobre una canción en la lista de canciones de un disco o busca canciones.
Historia de Usuario	HU-07
Resultado Esperado	Ver toda la letra original.
Estado	Éxito.

Tabla 3-56 PA Visualización de toda la Letra de una Canción

PA-16	
Nombre	Visualización de toda la letra traducida de una canción
Descripción	Se comprueba que se puede visualizar la letra traducida de la canción.
Procedimiento	▪ El usuario pulsa sobre traducción.
Historia de Usuario	HU-08
Resultado Esperado	Ver toda la letra traducida.
Estado	Éxito.

Tabla 3-57 PA Visualización de toda la Letra traducida de una Canción

PA-17	
Nombre	Visualización del resumen de una canción
Descripción	Se comprueba que se puede visualizar el resumen de la canción.
Procedimiento	▪ El usuario pulsa sobre resumen.
Historia de Usuario	HU-09
Resultado Esperado	Ver el resumen.
Estado	Éxito.

Tabla 3-58 PA Visualización del Resumen de una Canción

PA-18	
Nombre	Visualización del mensaje de una canción
Descripción	Se comprueba que se puede visualizar el mensaje de la canción.
Procedimiento	▪ El usuario pulsa sobre mensaje.
Historia de Usuario	HU-10
Resultado Esperado	Ver el mensaje.
Estado	Éxito.

Tabla 3-59 PA Visualización del Mensaje de una Canción

PA-19	
Nombre	Visualización de todo el TAB de una canción
Descripción	Se comprueba que se puede visualizar el tablature de la canción.
Procedimiento	▪ El usuario pulsa sobre TAB.
Historia de Usuario	HU-11
Resultado Esperado	Ver el TAB de la canción.
Estado	Éxito.

Tabla 3-60 PA Visualización de todo el TAB de una Canción

PA-20	
Nombre	Visualización de las noticias
Descripción	Se comprueba que se puede visualizar las noticias y los enlaces externos.
Procedimiento	▪ El usuario pulsa sobre noticias
Historia de Usuario	HU-03
Resultado Esperado	Ver las noticias.
Estado	Éxito.

Tabla 3-61 PA Visualización de las Noticias

PA-21	
Nombre	Visualización de las frases célebres
Descripción	Se comprueba que se puede visualizar las frases célebres y los enlaces externos.
Procedimiento	▪ El usuario pulsa sobre frases célebres.
Historia de Usuario	HU-02
Resultado Esperado	Ver las frases célebres.
Estado	Éxito.

Tabla 3-62 PA Visualización de las Frases Célebres

PA-22	
Nombre	Autenticación cono Admin o root
Descripción	Se comprueba que se puede acceder al menú de Admin o root según el usuario.
Procedimiento	▪ El usuario pulsa sobre Admin.
Historia de Usuario	HU-12
Resultado Esperado	Autenticarse en el sistema como Admin o root.
Estado	Éxito Si permite acceso sólo a usuarios autorizados.

Tabla 3-63 PA Autenticación como Admin o root

PA-23	
Nombre	Visualización menú Admin
Descripción	Se comprueba que se puede visualizar el menú de Admin.
Procedimiento	▪ El usuario accede al menú Admin.
Historia de Usuario	HU-13
Resultado Esperado	Ver el menú de Admin.
Estado	Éxito. Si sólo permite a usuarios que sean administradores.

Tabla 3-64 PA Visualización menú Admin

PA-24	
Nombre	Visualización menú root
Descripción	Se comprueba que se puede visualizar el menú de root.
Procedimiento	▪ El usuario pulsa sobre menú Admin.
Historia de Usuario	HU-14
Resultado Esperado	Ver el menú de root.
Estado	Éxito. Si sólo se permite a usuario root.

Tabla 3-65 PA Visualización menú Root

PA-25	
Nombre	Registrarse a uno o varios grupos
Descripción	Se comprueba que se puede registrar como usuario de uno o varios grupos para ser informado por correo al incorporar nuevos datos en el sistema.
Procedimiento	▪ El usuario pulsa registrarse o su nombre de usuario en el menú principal.
Historia de Usuario	HU-27
Resultado Esperado	Quedar registrado en el sistema.
Estado	Éxito. Si se almacena en el sistema y recibe correos al añadir información en el sistema.

Tabla 3-66 PA Registrarse a uno o varios Grupos

PA-26	
Nombre	Buscar desde menú principal
Descripción	Se comprueba que se puede buscar todas las canciones que contengan tal texto en su letra original.
Procedimiento	▪ El usuario pulsa sobre buscar.
Historia de Usuario	HU-30
Resultado Esperado	Ver un listado de todas las canciones que contengan en su letra original un texto dado.
Estado	Éxito.

Tabla 3-67 PA Buscar desde Menú Principal

PA-27	
Nombre	Buscar canciones con tal titulo
Descripción	Se comprueba que se puede buscar todas las canciones que contengan tal texto en su titulo.
Procedimiento	▪ El usuario pulsa sobre buscar.
Historia de Usuario	HU-31
Resultado Esperado	Ver un listado de todas las canciones que contengan en su título un texto dado.
Estado	Éxito.

Tabla 3-68 PA Buscar Canciones con tal Título

PA-28	
Nombre	Buscar grupos con tal nombre
Descripción	Se comprueba que se puede buscar todos los grupos que contengan tal texto en su nombre.
Procedimiento	▪ El usuario pulsa sobre buscar.
Historia de Usuario	HU-32
Resultado Esperado	Ver los grupos que cumplen criterio de búsqueda.
Estado	Éxito.

Tabla 3-69 PA Buscar Grupos con tal Nombre

PA-29	
Nombre	Buscar discos con tal nombre
Descripción	Se comprueba que se pueden buscar todos los discos que contengan tal texto en su titulo.
Procedimiento	▪ El usuario pulsa sobre buscar.
Historia de Usuario	HU-33
Resultado Esperado	Ver los discos que cumplen criterio de búsqueda.
Estado	Éxito.

Tabla 3-70 PA Buscar Discos con tal Nombre

PA-30	
Nombre	Cierre de sesión (log-out)
Descripción	Se comprueba que se puede cerrar la sesión con el Admin actual.
Procedimiento	▪ El usuario pulsa sobre Log-out.
Historia de Usuario	HU-25
Resultado Esperado	Se desconecta del usuario actual.
Estado	Éxito.

Tabla 3-71 PA Cierre de Sesión (Log-out)

PA-31	
Nombre	Cierre de sesión completo
Descripción	Se comprueba que se puede salir del todo del usuario autenticado.
Procedimiento	▪ Desde google.es pulsar sobre la imagen del usuario (arriba a la derecha) y sobre cerrar sesión.
Historia de Usuario	HU-26
Resultado Esperado	Ya no está en el sistema como el usuario anterior.
Estado	Éxito.

Tabla 3-72 PA Cierre de Sesión Completo

4. CAPÍTULO 4: GESTIÓN DEL PROYECTO

En este capítulo se proporcionan los costes de la realización del proyecto y la planificación para su desarrollo.

4.1 PLANIFICACIÓN

Para el desarrollo del proyecto se ha utilizado la metodología Scrum, por lo que éste ha sido planificado en Sprint. El proyecto ha sido realizado en cuatro Sprint, de quince días cada uno sin contar con los fines de semana y festivos, durante los cuales se han realizado una serie de Historias de Usuario.

Antes de realizar el primer Sprint, se analizaron las tecnologías cliente y servidor con el fin de elegir la más adecuada, también se describieron las Historias de Usuario, se definió la base de datos y se realizó un prototipo de la aplicación. A continuación se muestra la planificación de los Sprints. Cada historia de usuario realizada incluye sus pruebas de aceptación.

Sprint 1
Del 8 de Junio al 26 de Junio.
1

Sprint 2
Del 29 de Junio al 17 de Julio.
2

Sprint 3
Del 20 de Julio al 7 de Agosto
3

Sprint 4
Del 10 de Agosto al 28 de Agosto
4

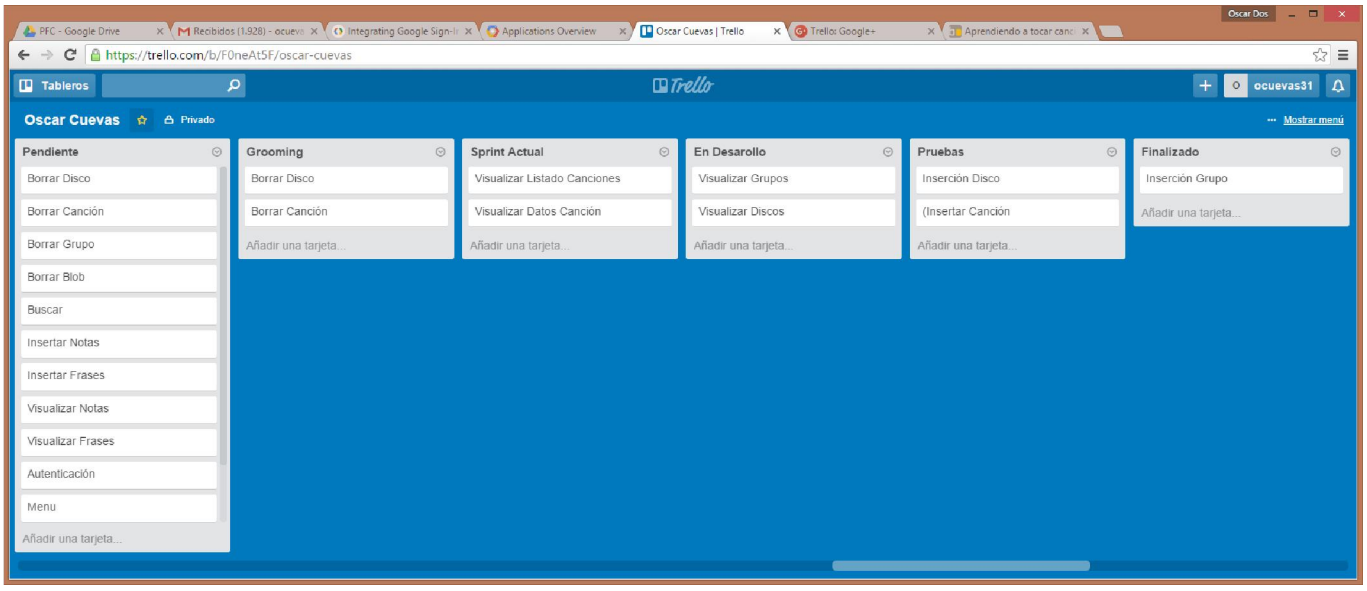


Ilustración 4-1 Captura de Trello para el Sprint 2

- **Sprint 1**: Durante este Sprint se llevaron a cabo las historias de usuario HU-15 (Inserción Grupo), HU-17 (Inserción Disco), HU-19(Insertar Canción).
  Al final del Sprint se le entregó al Product Owner una aplicación donde podía ver la página de Inserción de la aplicación.

- **Sprint 2**: Durante el Sprint 2 se realizaron las historias de usuario HU-04 (Visualizar Grupos), HU-05 (Visualizar Discos), HU-06 (Visualizar Listado Canciones), HU-07 (Visualizar Datos Canción), HU-08 (Visualizar Letra Canción), HU-09 (Visualizar resumen), HU-10 (Visualizar mensaje), HU-11 (Visualizar TAB).
  Al final del Sprint el Product Owner podía visualizar todos los datos de grupos, discos y canciones.

- **Sprint 3**: Durante este Sprint se han desarrollado las historias de usuario HU-21 (Borrar Disco), HU-22 (Borrar Canción), HU-28 (Borrar Grupo), HU-29 (Borrar Blob), HU-30 (Buscar Canción Letra), HU-31 (Buscar Canción Título), HU-32 (Buscar Grupo), HU-33 (Buscar Disco).
  Al final del Sprint el Product Owner es capaz de Borrar datos de la aplicación. También podrá realizar búsquedas.

- **Sprint 4**: Durante el último Sprint se desarrollo las historias de usuario HU-23 (Insertar Notas), HU-24 (Insertar Frases), HU-02 (Visualizar Notas), HU-03 (Visualizar Frases), HU-12 (Autenticación), HU-13 (Menú Admin), HU-14 (Menú Root), HU-01 (Portada), HU-25 (log-out), HU-26 (log-out completo) y HU-27 (Registrarse).
  Al final de este Sprint el Product Owner obtuvo el proyecto completo.

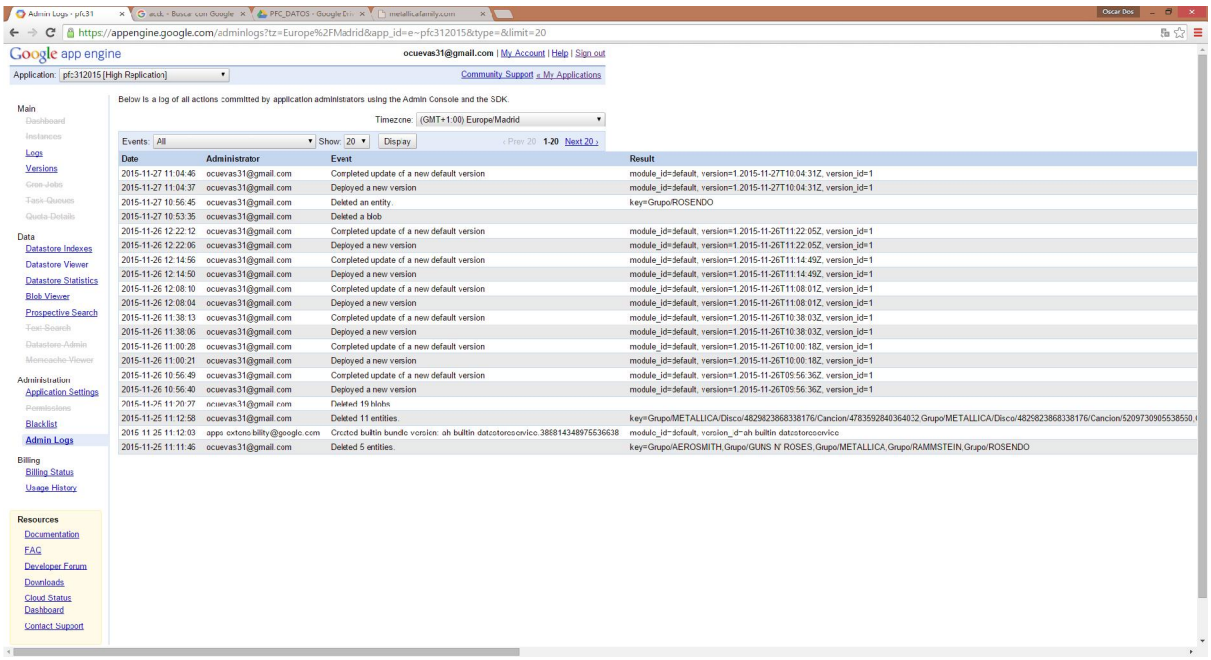
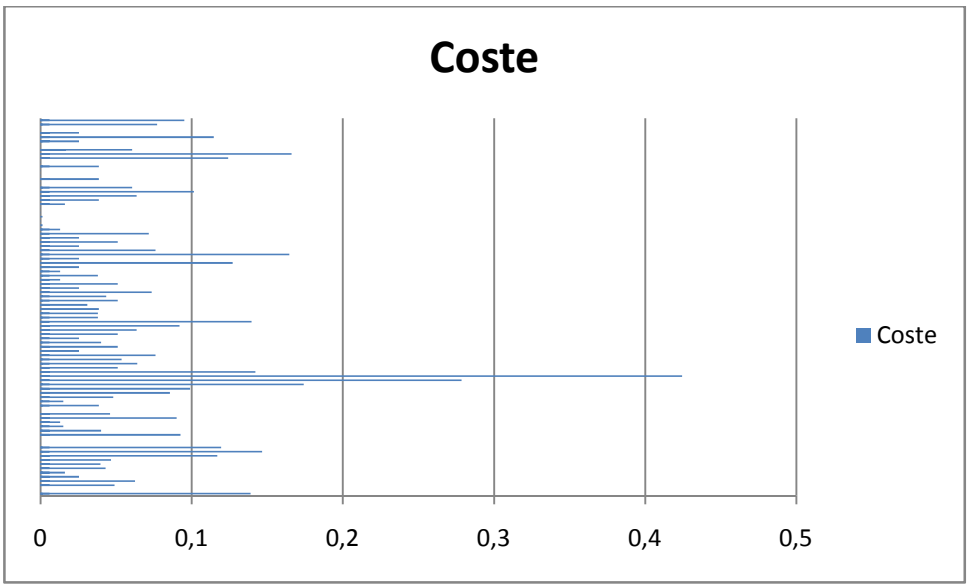


Ilustración 4-2 Ejemplo de Logs de eventos realizados en GAE. Ejemplo de un Sprint[xlviii].

4.2 PRESUPUESTO

El siguiente gráfico muestra el coste que habría que pagar a Google por su servicio GAE, pero como la cuenta incluye unos límites en su cuenta gratis, no se han tenido que abonar; si nos pasásemos de esos límites si se tendrían que abonar.



Total: 6,26788 €

Este gráfico está realizado con una tabla con 3.000 entradas, disponible desde la consola de GAE[xlix].

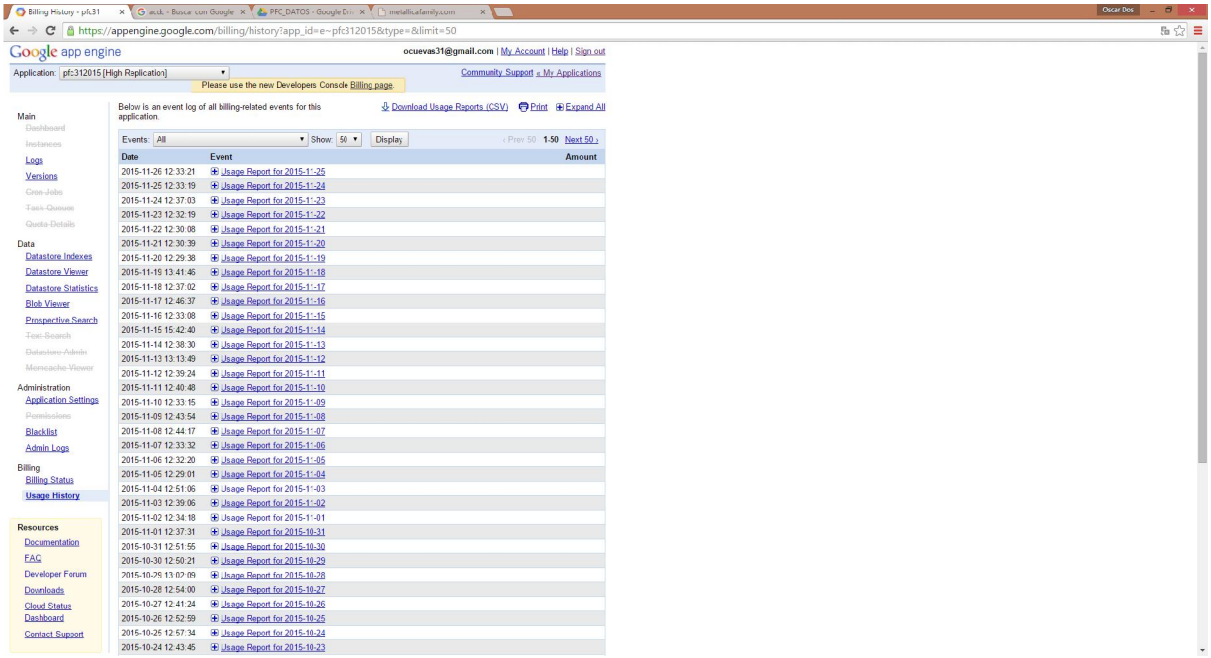


Ilustración 4-3 Resumen de Gastos de GAE

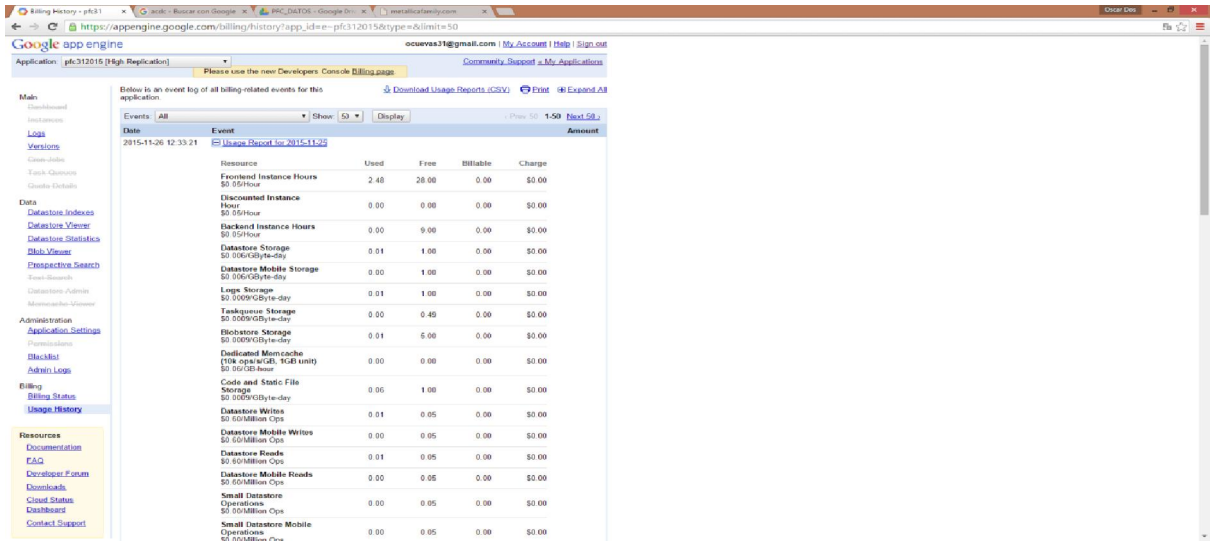


Ilustración 4-4 Detalle Gastos de GAE

El coste total de desarrollar el proyecto es OCHO MIL NOVECIENTOS CUARENTA Y SIETE EUROS CON NOVENTA Y CUATRO CÉNTIMOS DE EURO.

Desglose del presupuesto (Costes Directos):

Personal

En primer lugar se tendrán en cuenta los gastos asociados al personal que ha intervenido en el desarrollo del proyecto, teniendo en cuenta las siguientes consideraciones:

- Total días del proyecto = 93 días (3 meses)
- Horas al día de dedicación al proyecto = 5 horas
- Total de horas dedicadas del Scrum Manager y Product Owner = 45
- Dedicación hombre/mes = 155 horas
- Coste hombre/mes (Ingeniero Júnior) = 2694,39€
- Coste hombre/mes (Ingeniero Sénior, Product Owner y Scrum Master) = 4.289,54€
- Coste hombre/hora (Ingeniero Júnior): 11,23€
- Coste hombre/hora (Ingeniero Sénior): 17,87€

Apellidos y Nombre	Categoría	Dedicación (horas/mes)	Meses	Coste (Hombre/Hora)	Coste (Hombre/mes)	Coste Total
Hernando Corrochano, Jesús	Ingeniero Sénior	15	3	17,87 €	4.289,54 €	804,29 €
Cuevas Lanchares, Óscar	Ingeniero Júnior	155	3	11,23 €	2.694,39 €	5.220,38 €
Total						**6.024,67 €**

Tabla 4-1 Costes Personal

Para realizar el coste hombre/hora se ha dividido el coste del hombre/mes entre ocho, que son las horas de una jornada laboral normal. Para calcular el coste personal total mensual se han multiplicado el número de horas mensuales por el número de meses y por el coste hombre/hora. Siguiendo la siguiente fórmula:

$$Coste\ Personal = \sum_{b}^{a} Dedicación * Meses * Coste\ \left(\frac{h}{h}\right)$$

Donde:

a = Equipo Desarrollo (Ingenieros Júnior)
b = Ingenieros Sénior Scrum Máster y Producto Owner

La tabla 4.1 recoge estos costes. Como resultado obtenemos que los costes del personal ascienden a un total de SEIS MIL VEINTICUATRO EUROS CON SESENTA Y SIETE CÉNTIMOS DE EURO.

Equipos

Para la realización del proyecto se han necesitado los siguientes equipos:

- Portátil marca ACER Aspire V 17 Nitro Black Edition. 16 GB RAM. Disco SSD 250GB. Disco duro 2TB. Gráfica Nvidia, 1.237 €
- Smartphone SAMSUMNG GRAND 2. Adquirido con 6 meses de anterioridad, con un coste de 318,75€

La siguiente tabla recoge el coste de los equipos utilizados sin IVA, el uso dedicado el proyecto, su dedicación en meses y su periodo de depreciación.

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
ACER Aspire V 17 Nitro Black Edition.	1.237 €	100	3	60	121,88 €
Smartphone SAMSUNG GRAND 2	318,75 €	100	3	60	15,94 €
Total					**137,82 €**

Tabla 4-2 Costes Equipos

La fórmula utilizada para el cálculo de la amortización de equipos es la siguiente:

$$Amortización = \frac{A}{B} * C * D$$

Donde:

A = Número de meses desde la fecha de facturación en que el equipo es utilizado.
B = Periodo de depreciación (60 meses).
C = Coste del equipo (sin IVA).
D = % del uso que se dedica al proyecto.

Como resultado hemos obtenido que el coste de la amortización de los equipos utilizados durante el proyecto es de CIENTO TREINTA Y SIETE EUROS CON OCHENTA Y DOS CÉNTIMOS DE EURO.

Resumen de Costes

La siguiente tabla recoge el resumen de los costes aplicables al proyecto. Los costes Indirectos corresponderán al 20%.

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	6.024,67 €
Amortización	137,82 €
Costes Indirectos	1.232,50 €
Total (sin IVA)	**7.394,99 €**
Total (IVA)	**8.947,94 €**

Tabla 4-3 Resumen Costes Proyecto

El coste total de desarrollar el proyecto es OCHO MIL NOVECIENTOS CUARENTA Y SIETE EUROS CON NOVENTA Y CUATRO CÉNTIMOS DE EURO.

5. CAPÍTULO 5: CONCLUSIONES Y LÍNEAS FUTURAS

He tenido que realizar un trabajo previo de investigación y actualización muy grande, ya que termine la carrera de I.T.I.G. en 1997 y no sabía nada de JavaScript, CSS, servlets, jsp, persistencia de datos en la nube, interfaz de usuario, etc. y muy poco de HTML5. Además, en concreto, de persistencia en GAE sólo existe la documentación de Google (en inglés) y muy pocos ejemplos, por lo que la única opción era probar, es decir realizar un importante trabajo de laboratorio. El crear las entidades y, sobre todo, las claves (keys) ha sido un proceso bastante tedioso, aunque como todo, una vez conseguido reconforta.

Los errores de sintaxis, las comillas, concatenaciones, etc. hacen perder mucho tiempo, al no tener experiencia en los nuevos lenguajes y mezclarlos todos juntos en el código. Por ejemplo, mezclar Java, Html, JavaScript, jsp, servlets.

Durante el desarrollo del PFC he tenido que hacer un importante trabajo adaptativo, ya que google está haciendo cambios y mejoras en su GAE por lo que me he tenido que ir adaptando a ellos e ir utilizando sus nuevas funcionalidades y modificando las que cambiaban.

Lecciones positivas:

- Saber “frenar los pies” a los requisitos propuestos por el cliente. En este caso concreto, ha sido a las historias de usuario. El cliente siempre va a querer tener la mejor aplicación, y su mente está en continua lluvia de ideas para lanzar a los responsables del sistema sus pensamientos y características nuevas a añadir, y que quiere que estén para la primera versión de despliegue.

- Adopción de diferentes mentalidades en la realización del proyecto. La realización de un proyecto de tal calibre como es un sistema servidor con diferentes clientes, ha supuesto todo un reto, ya que no puedes pensar solamente en cómo desarrollar el sistema para que funcione, sino que tienes que pensar en cómo desarrollar un sistema que funcione y en el que los usuarios se sientan cómodos con la interfaz, así como que se visualice de la misma forma en los diversos dispositivos. También hay que lograr que sea seguro para proteger la confidencialidad de los usuarios, etc., es decir, son un cúmulo de finalidades que se tienen que pensar a la hora de desarrollar el sistema y que han servido como punto de partida de los pensamientos a seguir a la hora de afrontar los proyectos que vendrán en el ámbito profesional.

- Se han perfeccionado conocimientos de seguridad, de desarrollo de aplicaciones, de interfaces de usuario y de gestión de proyectos. La motivación con la que se empezó el proyecto para afianzar los conocimientos de un grupo de asignaturas de la carrera ha repercutido positivamente.

Lecciones negativas:

- La metodología Scrum no es apta para equipos de trabajos de dos personas. Al usar una metodología de desarrollo ágil como ha sido Scrum, se ha podido comprobar de primera mano cómo esta metodología fue creada para gestionar un equipo de desarrollo de varias personas.
- Los grandes proyectos no pueden realizarse por una persona. La realización de un gran proyecto, como ha sido el ejecutado, podría haberse terminado en un plazo menor de tiempo si se hubiese realizado por un conjunto mayor de personas, que estuviesen motivadas y poseyeran los conocimientos necesarios.

- Para realizar un proyecto medianamente grande, no es una buena idea tener que aprender nuevas tecnologías/herramientas, es mucho mejor utilizar las ya ampliamente conocidas y con muchos ejemplos y documentación, ya que complica bastante el tener que investigar cómo hacer las cosas y cuál es la mejor forma.

Líneas futuras:

Creo que sería muy interesante poder ampliar el proyecto para incorporar cualquier tipo de información, de hecho el sistema está realizado de la forma más genérica posible para poder ampliar a otros tipos de información, como libros, cuadros, etc. En lugar de grupos, serían escritores y ,en lugar de discos, libros, etc.

Se podrían realizar bots[l] (robots) que buscasen información en Internet sobre algún grupo musical, como discos, noticias, traducciones de letras, etc. y que mandase un correo al Admin de ese grupo para que éste diese su permiso para incorporar esa información en la aplicación, previa revisión por él mismo de tal información.

Pensando en explotar la aplicación desde un sentido económico sería interesante incorporar publicidad a la misma, cosa que no me sería nada complicado ya que soy miembro del programa Adsense[li] de Google y tengo alguna aplicación que ya contiene publicidad como por ejemplo http://nosuspendo.com.

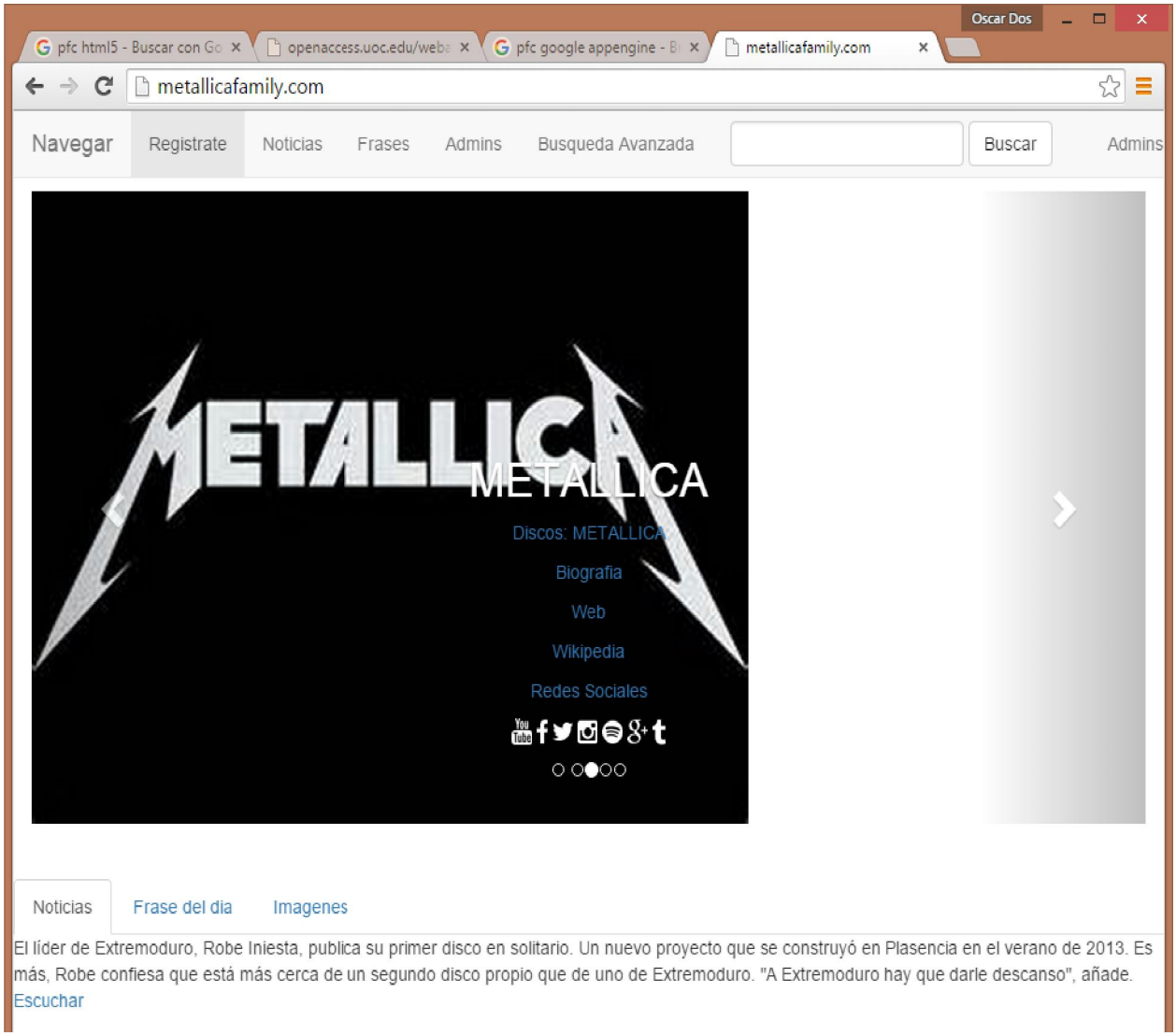
Algo básico como línea futura podría ser interesarse mucho más por la UX[lii] (Experiencia de Usuario), ya que hoy en día es muy importante no sólo la información que contenga una aplicación, sino su imagen externa. Además, esto me permitiría ampliar los conocimientos adquiridos durante la ejecución del PFC.

También se podría aumentar la seguridad de la aplicación, aunque no es algo prioritario y es algo que siempre se puede hacer en todas las aplicaciones.

ANEXO 1: MANUAL DE USUARIO

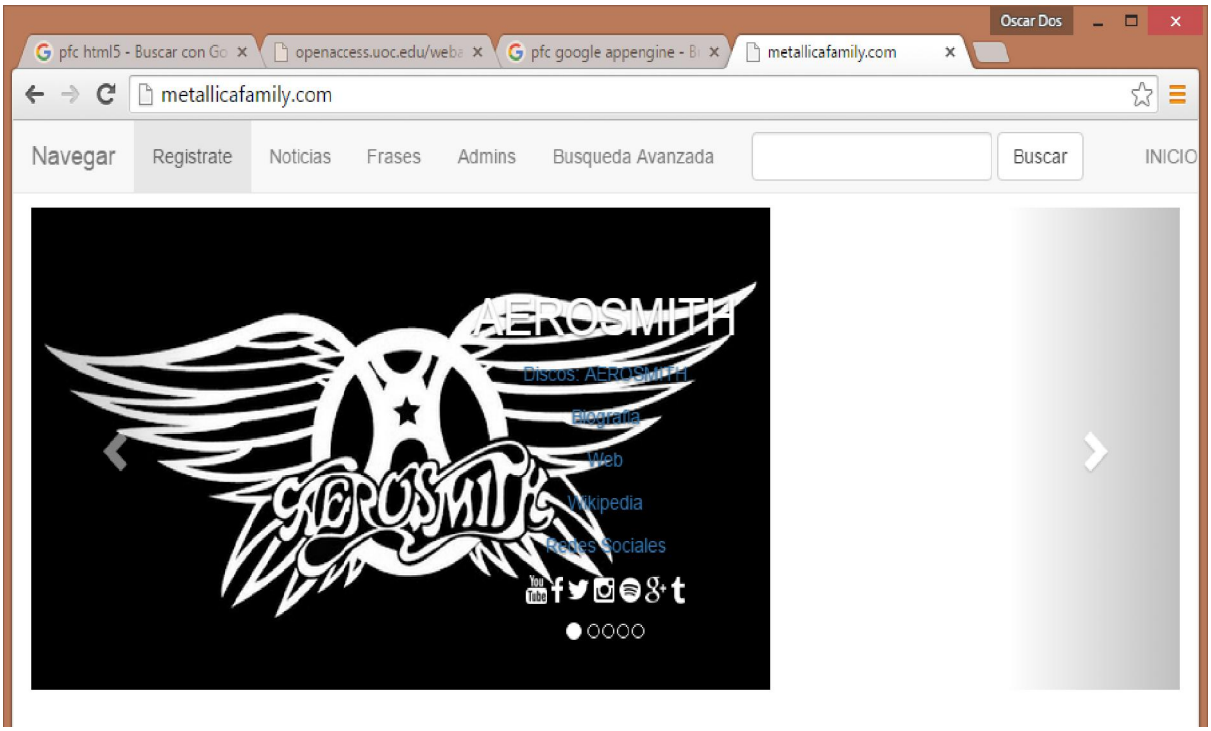
En este anexo se incluye un manual de uso de la aplicación.

Inicio de la Aplicación:

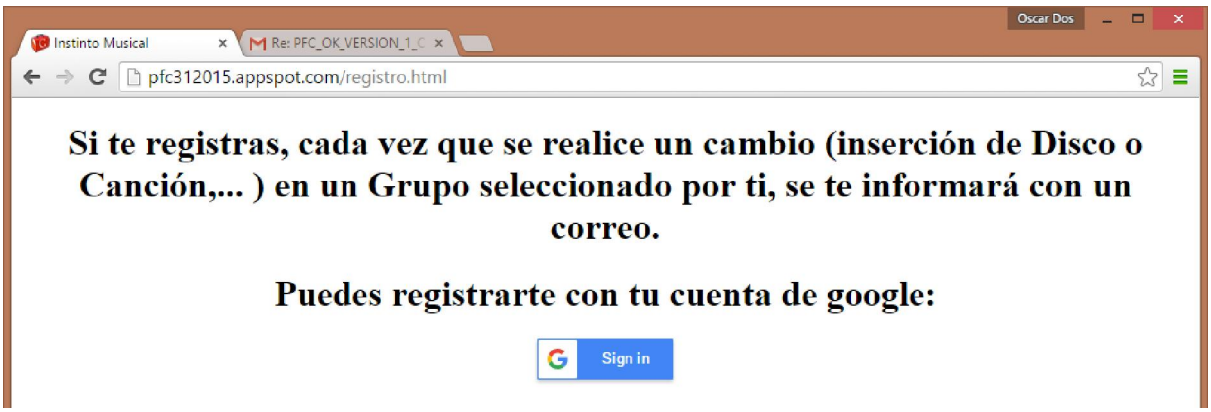


Se muestra al inicio de la aplicación los grupos en forma de carrusel, automático o mediante botones.

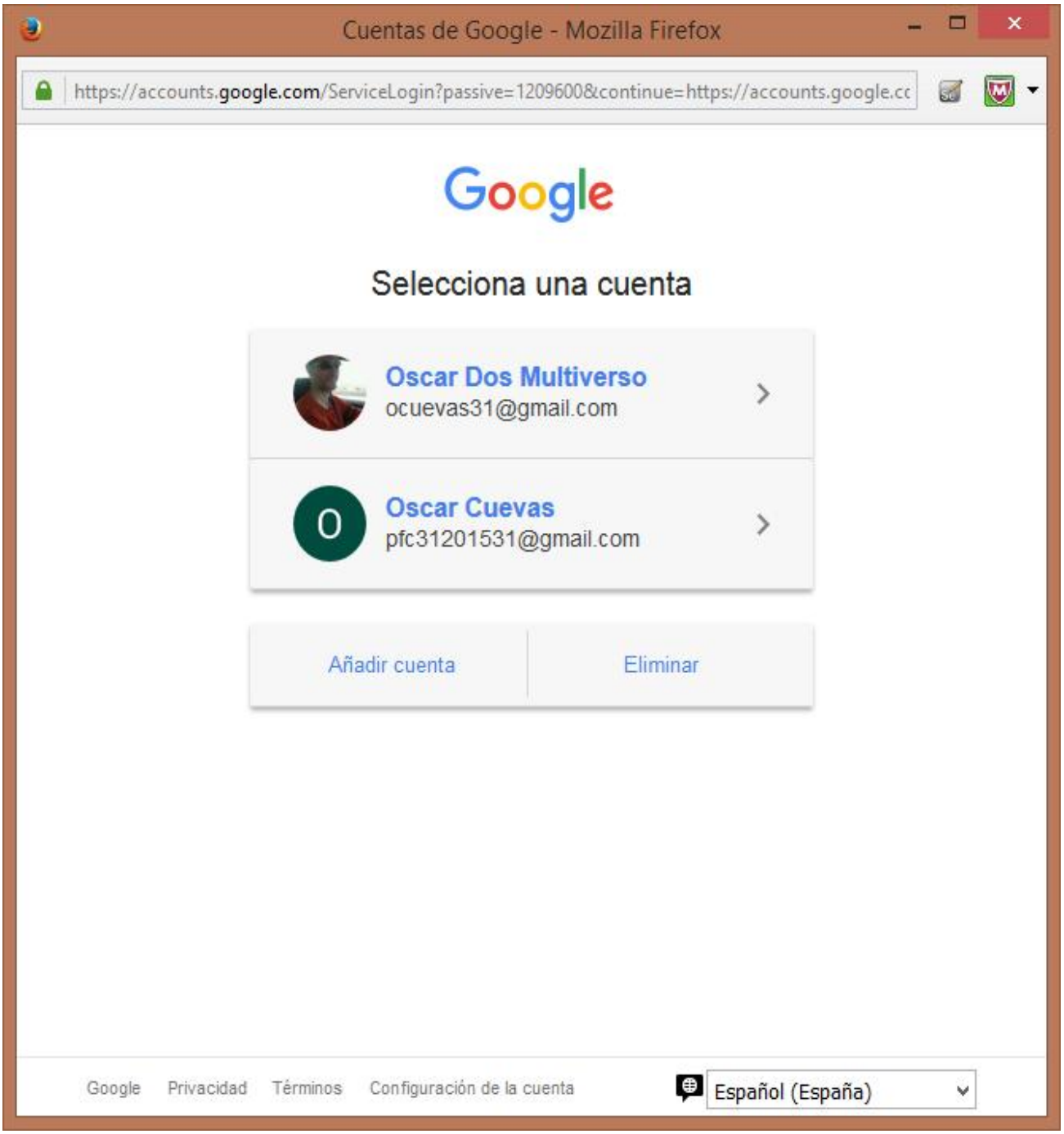
Registrarse:



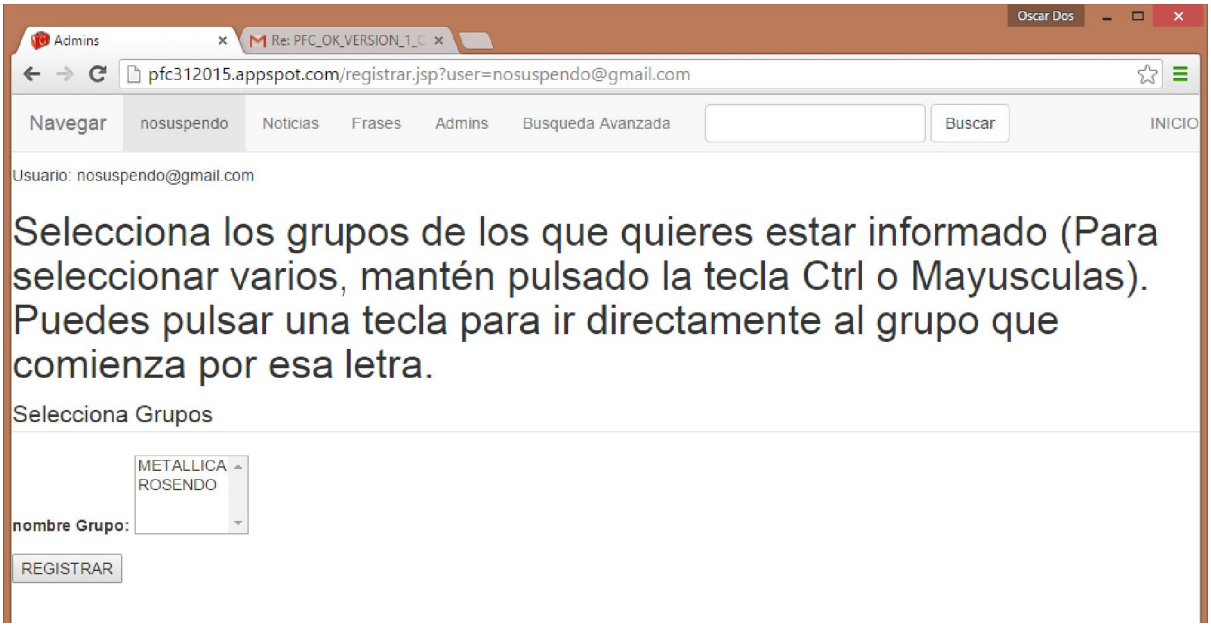
Se puede pulsar sobre registrarse del menú. Aparece la ventana de registro, mediante una cuenta de Google:



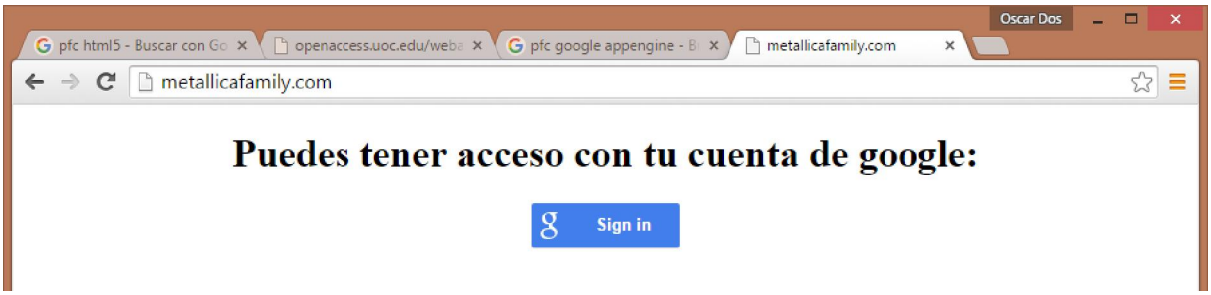
Una vez se selecciona la cuenta:



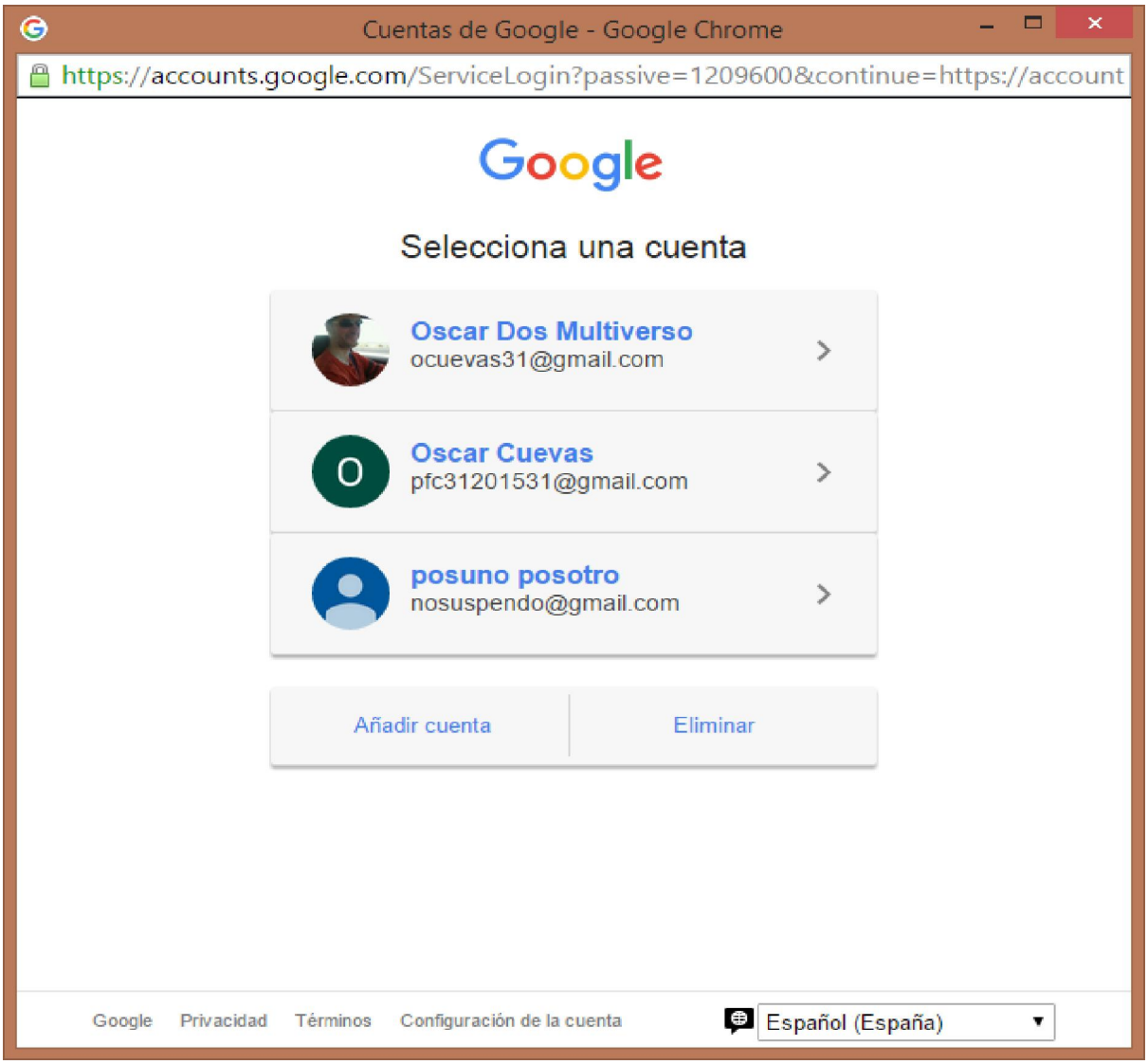
Ahora se pueden seleccionar los grupos en los que registrarse para recibir un correo para ser informado de cualquier modificación que se realice de cualquiera de esos grupos:



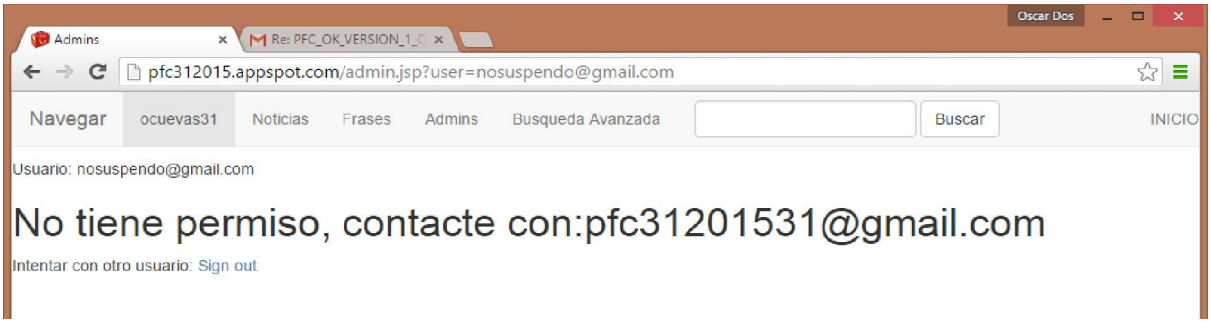
Los administradores y root pueden insertar y borrar los datos que están en la aplicación, para ello pulsan sobre Admin en el menú, y se procede al acceso seguro mediante una cuenta de Google de Gmail:



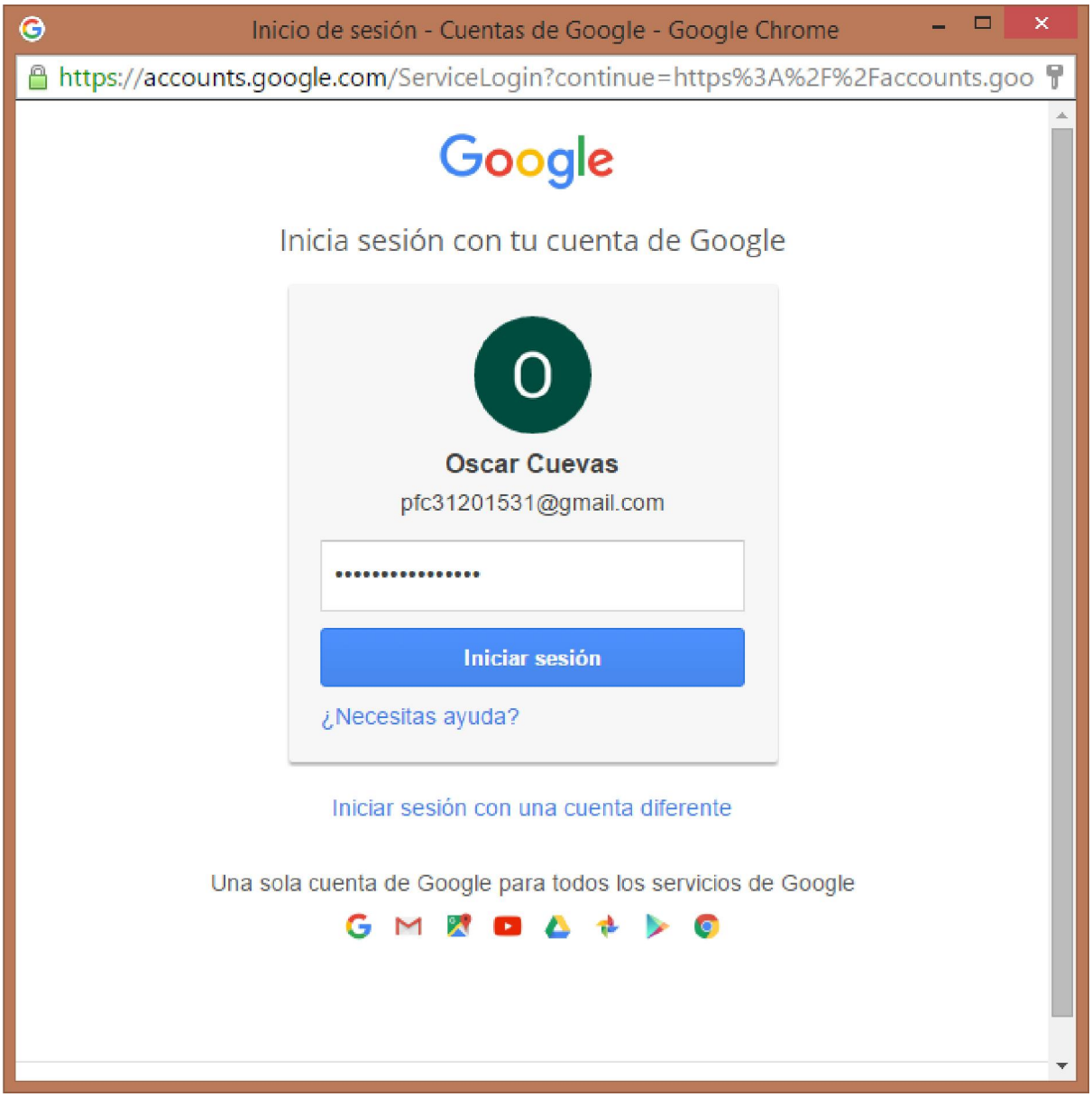
Si tienes varias cuentas de Gmail de Google, selecciona cuál:



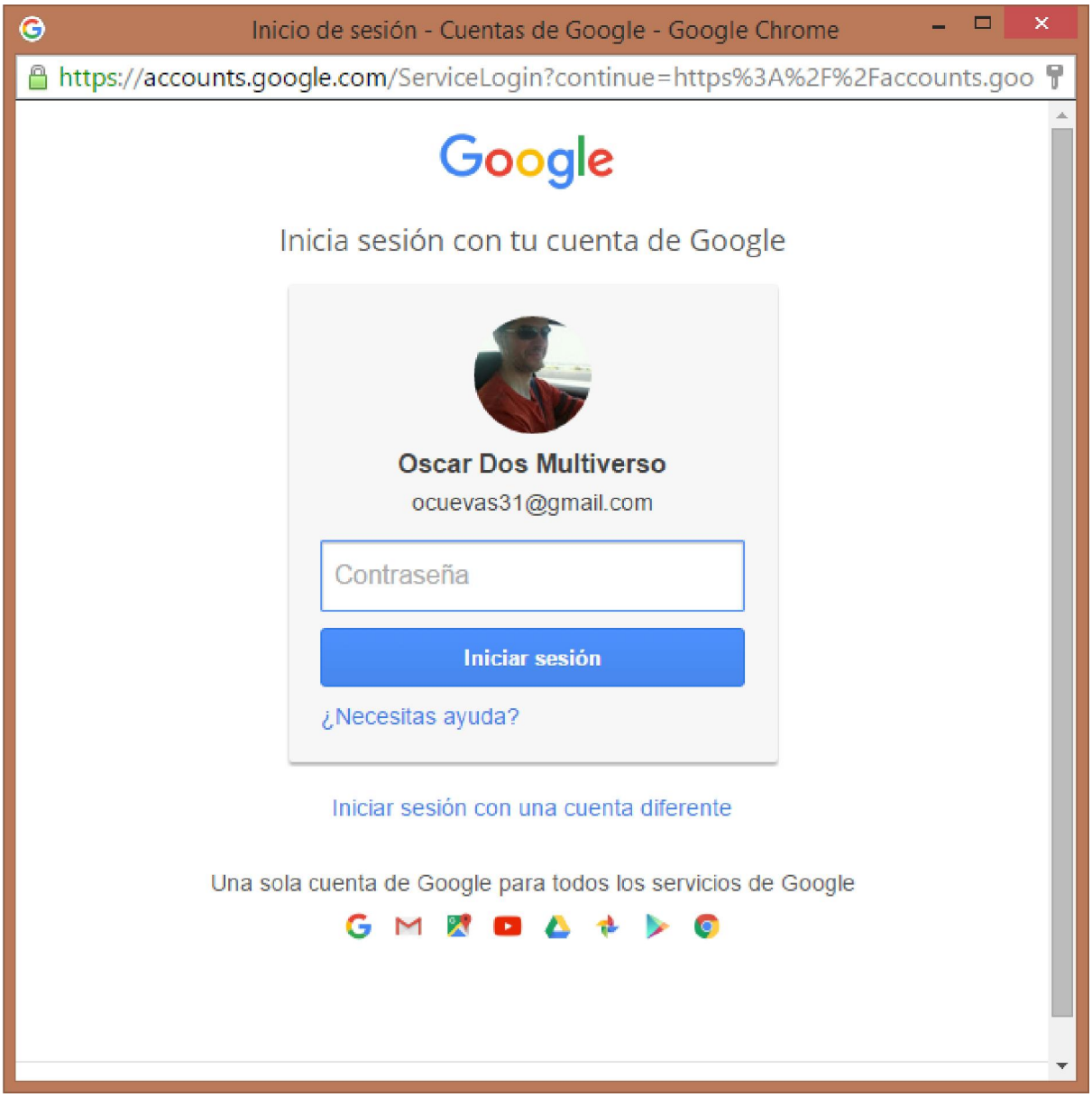
Pantalla de acceso no concedido. El usuario no es un administrador:



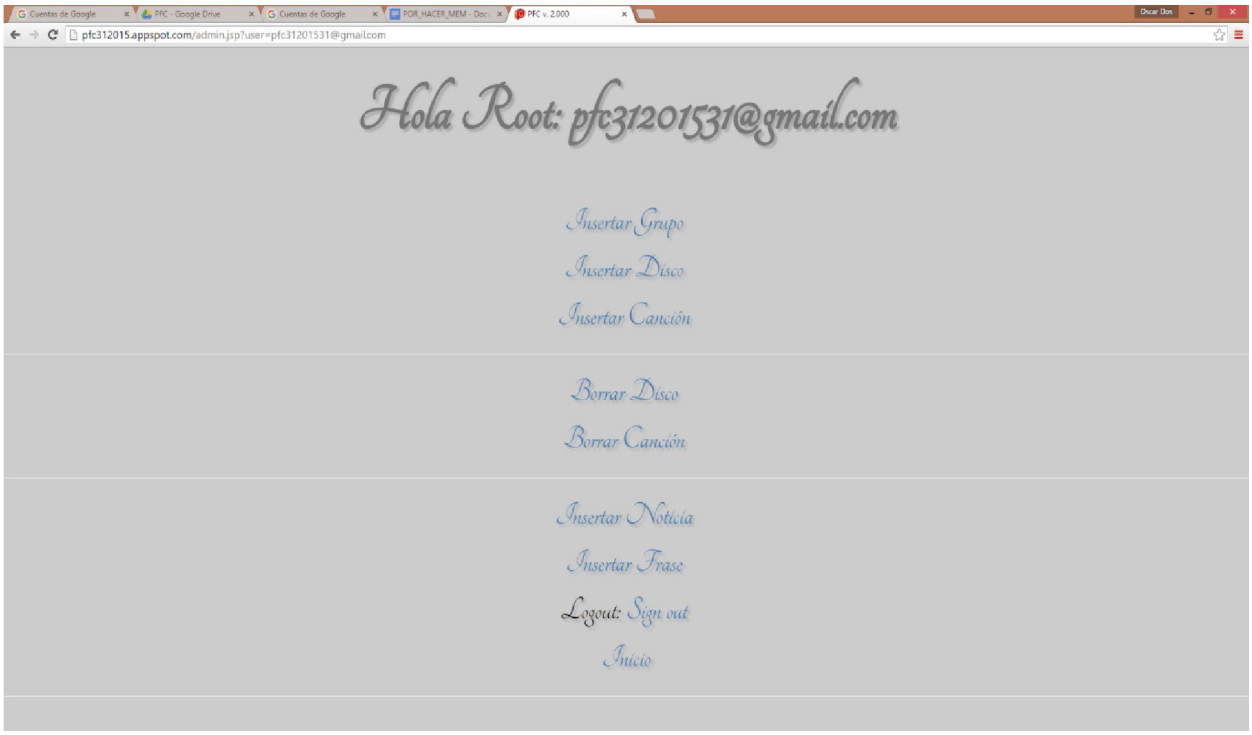
Pantalla de acceso como root:



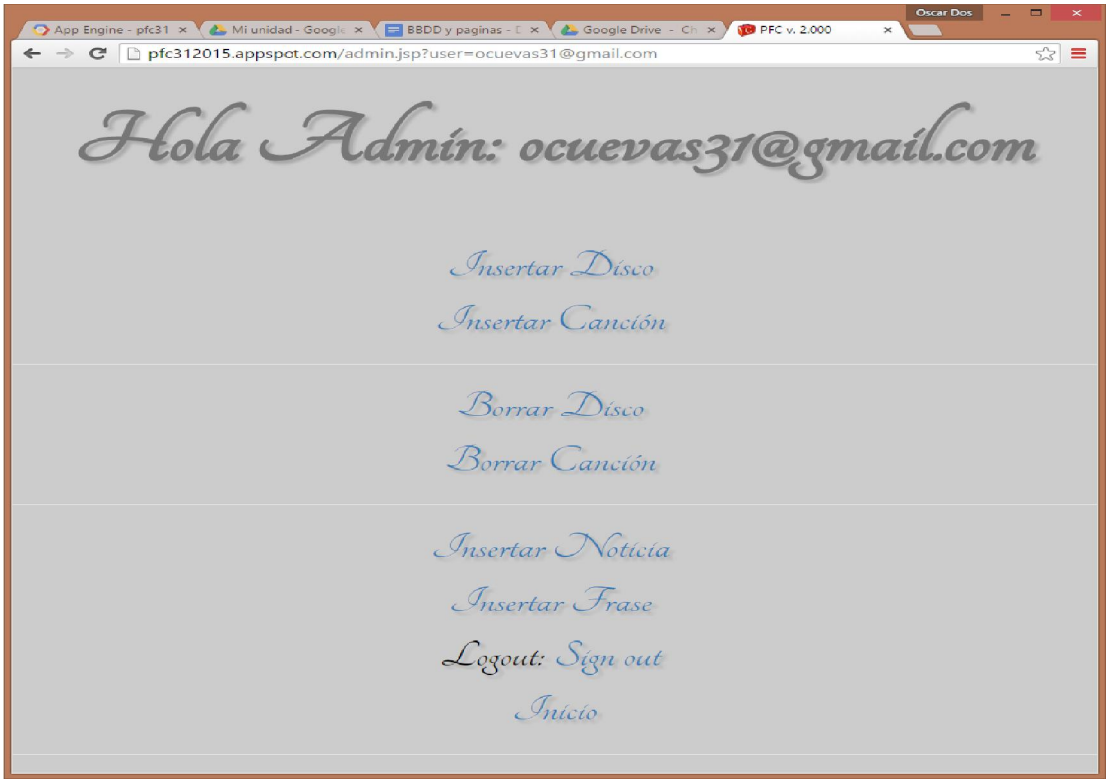
Pantalla de acceso de administrador:



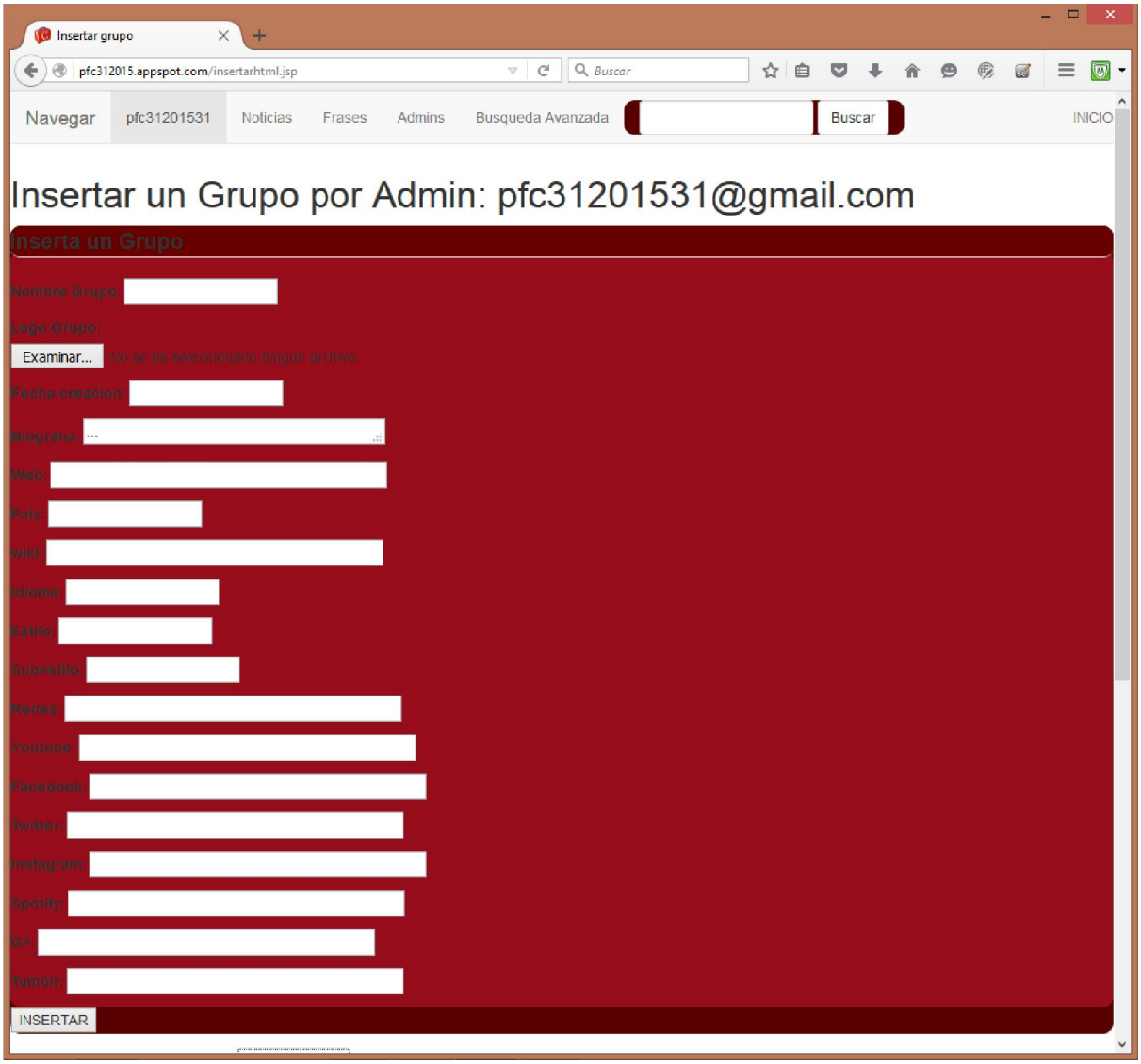
Acceso concedido como root y menú:



Acceso concedido como Admin:

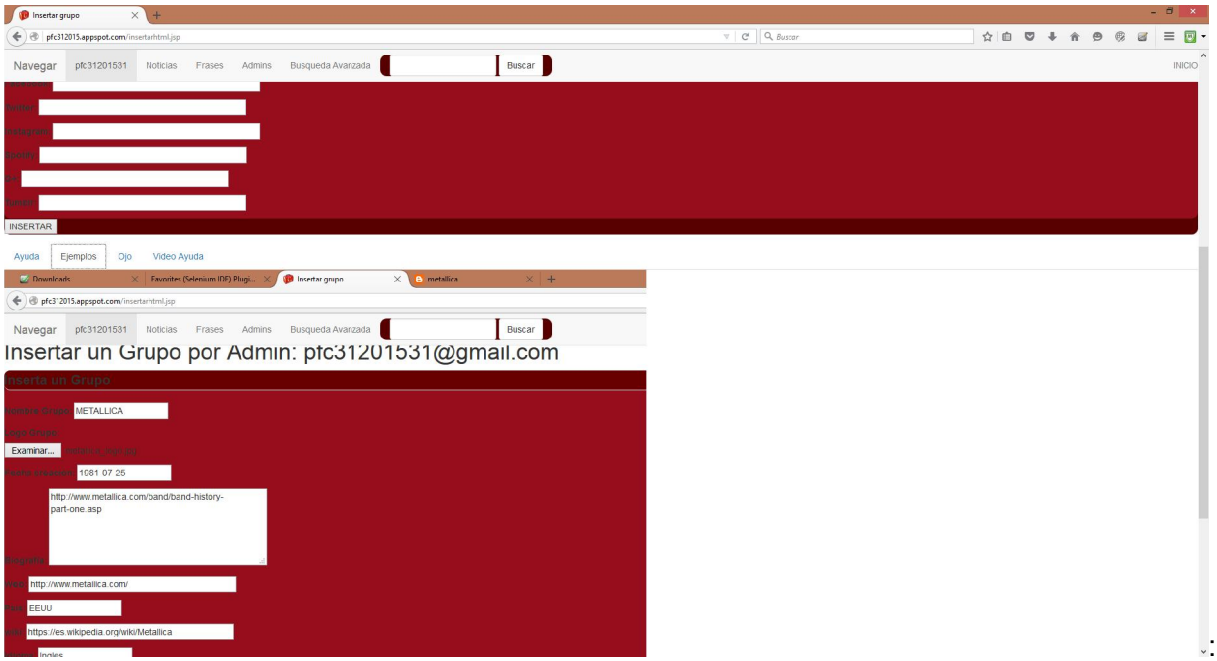


Insertar Grupo:



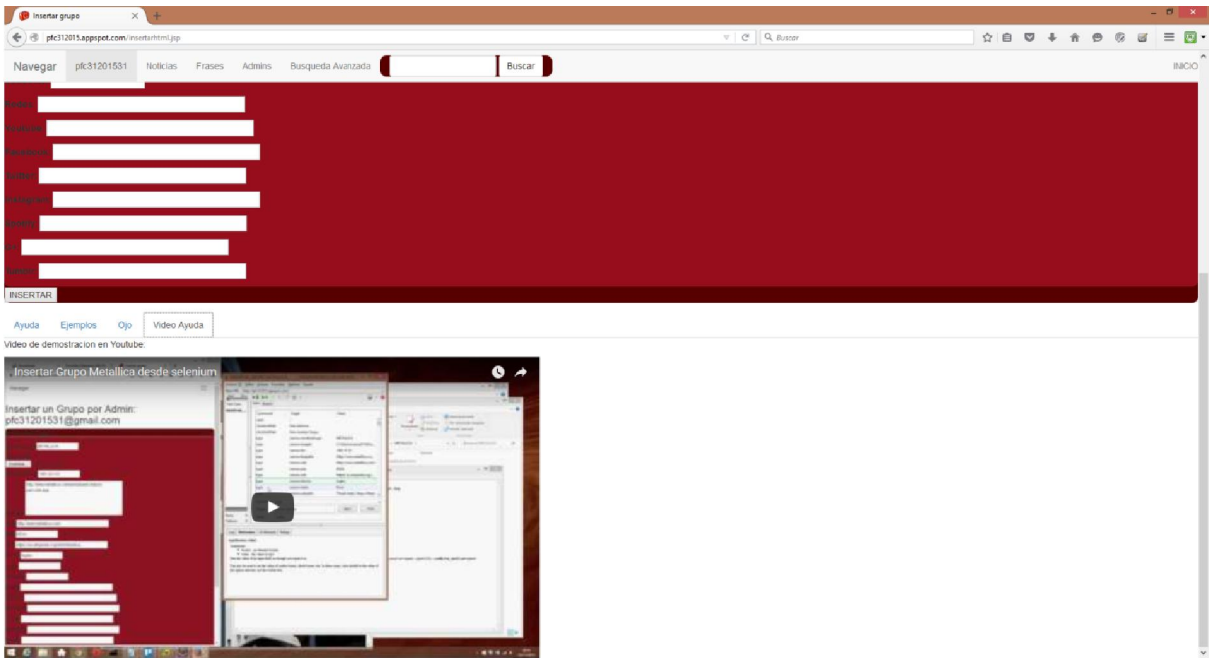
Video de Ayuda insertar Grupo:
https://www.youtube.com/watch?v=CMByzXJX27c

Pestaña Ejemplos de insertar un Grupo:

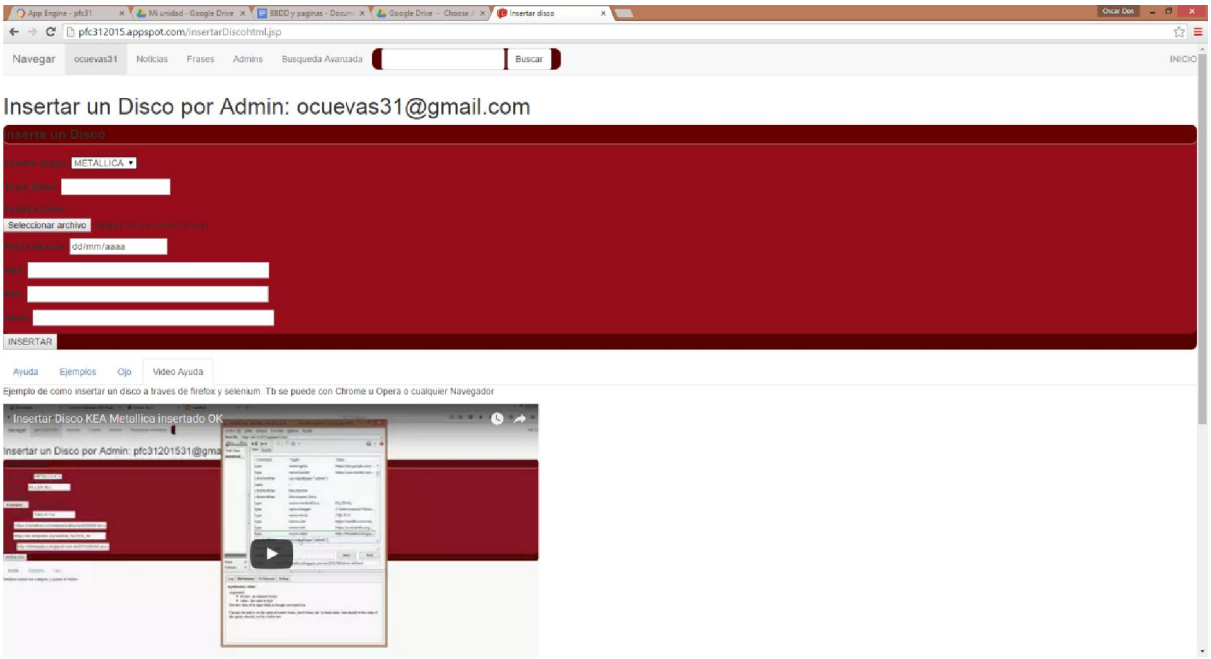


En las pestañas de Ayuda y Ojo aparecen indicaciones y cosas a tener en cuenta.

Pestaña de vídeo Ayuda:



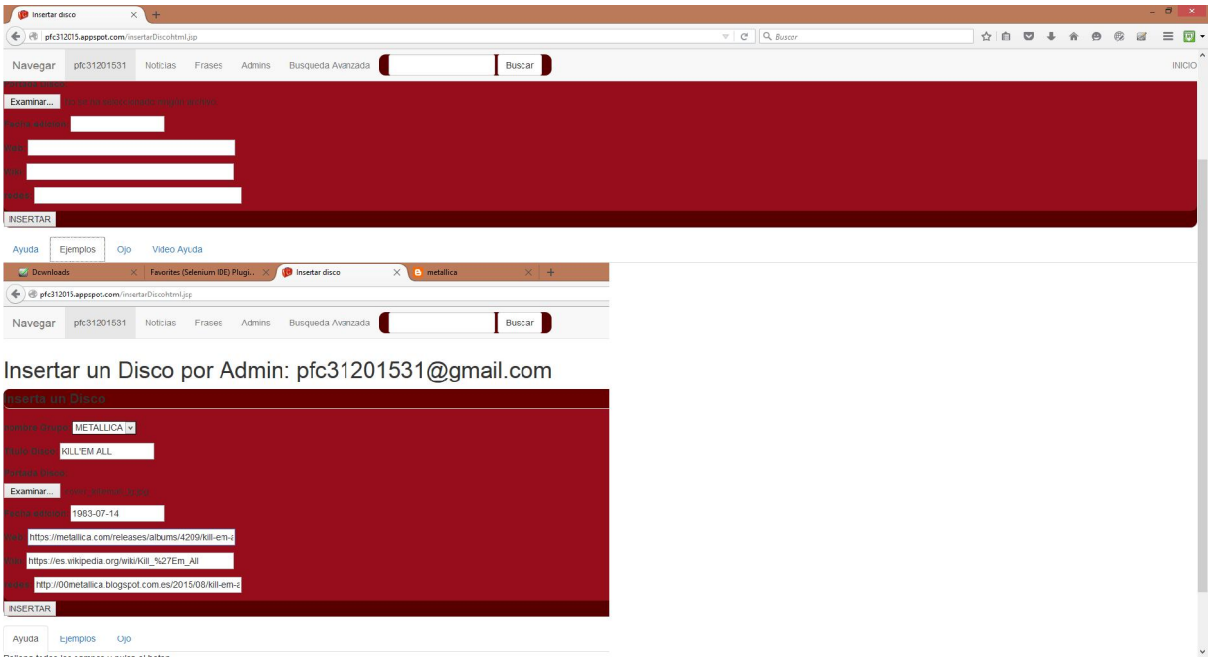
Insertar Disco:



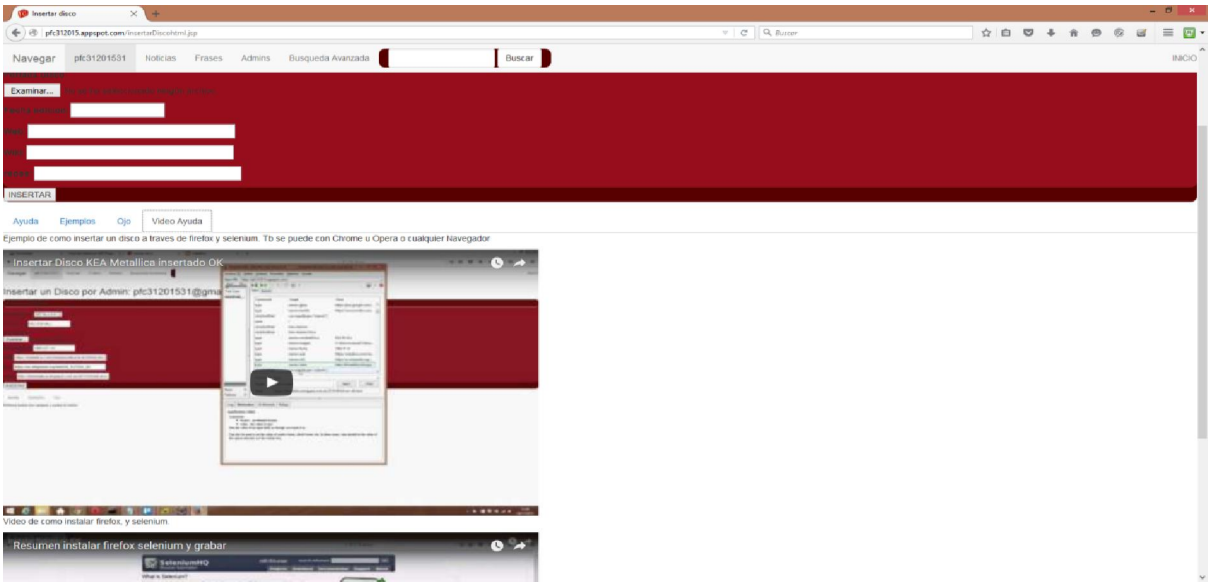
Video de Ayuda insertar un Disco:
https://www.youtube.com/watch?v=JUsXu0ToC2M

En las pestañas de Ayuda y Ojo aparecen indicaciones y cosas a tener en cuenta.

Insertar Disco, pestaña Ejemplos:



Insertar Disco, pestaña de Vídeo Ayuda:



Insertar Canción:

Vídeo de Ejemplo de cómo insertar una Canción:
https://www.youtube.com/embed/5R1E8dFYCUw

Vídeo de Ejemplo de cómo insertar Canciones:
https://youtu.be/E7U1bxG-mxw

Vídeo de Ejemplo de cómo insertar Discos y navegar por ellos y por Canciones.
https://youtu.be/0eci9g7GQr0

Al insertar una canción se envía un correo, tanto al root como a los usuarios registrados:

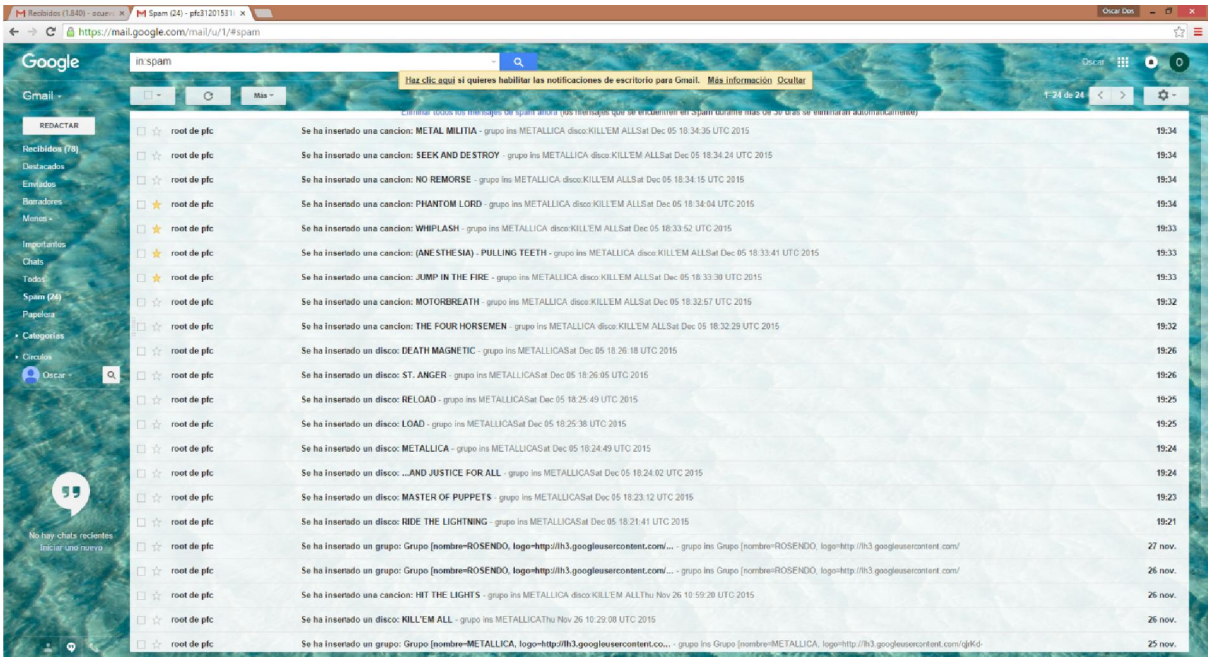
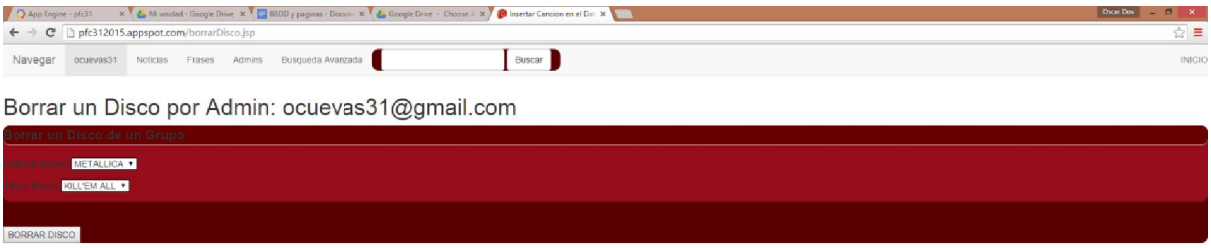
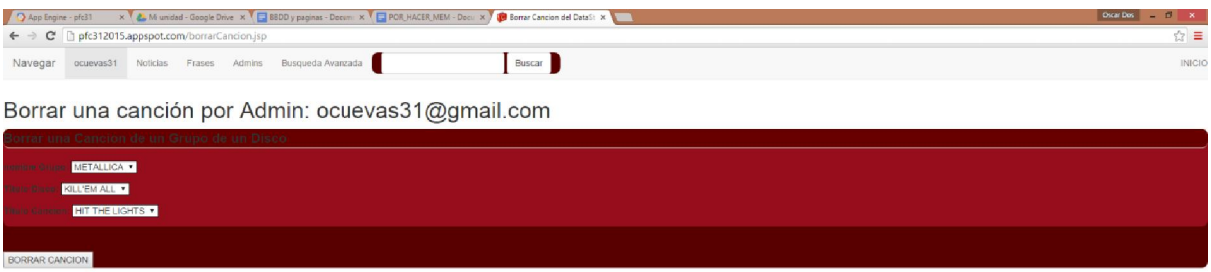


Ilustración 5-1 Correos Enviados Automáticamente por la Aplicación al Root.

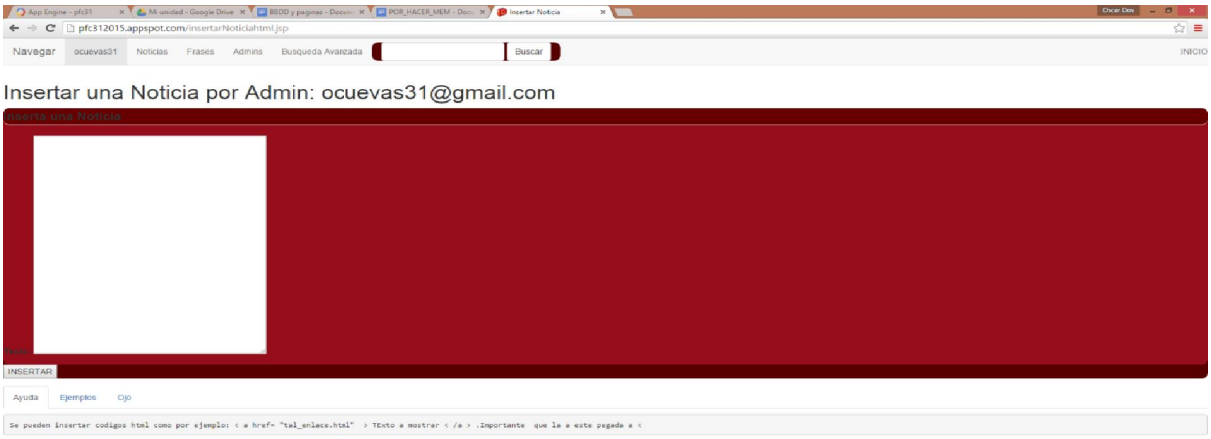
Borrar Disco:



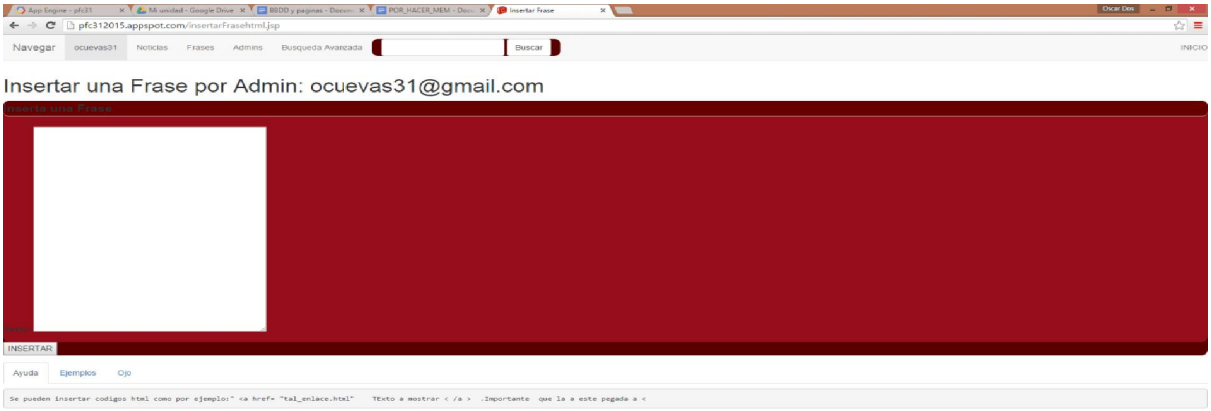
Borrar Canción:



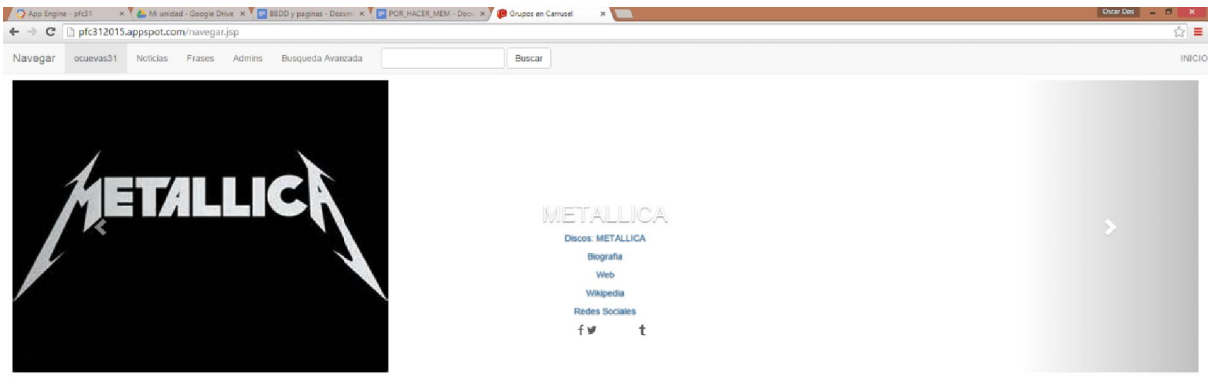
Insertar Noticia:



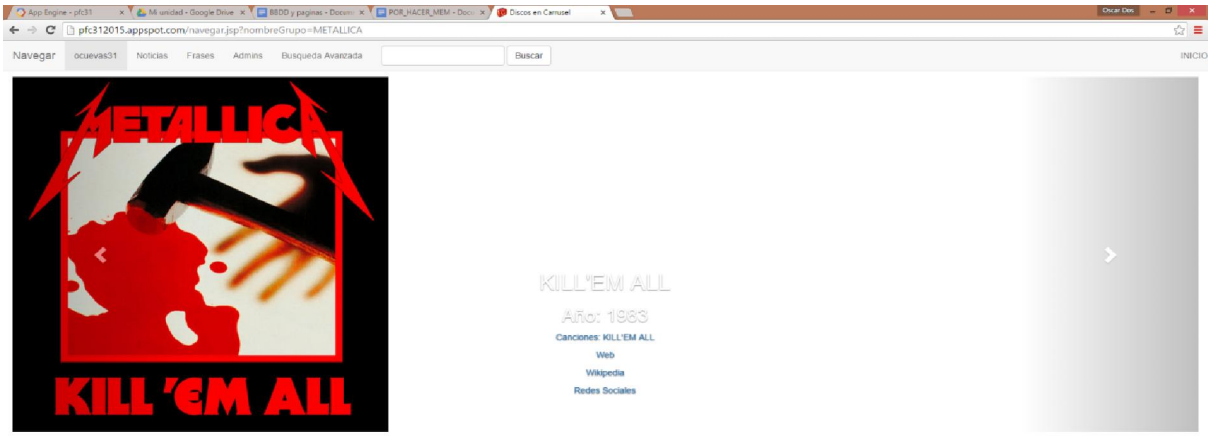
Insertar Frase:



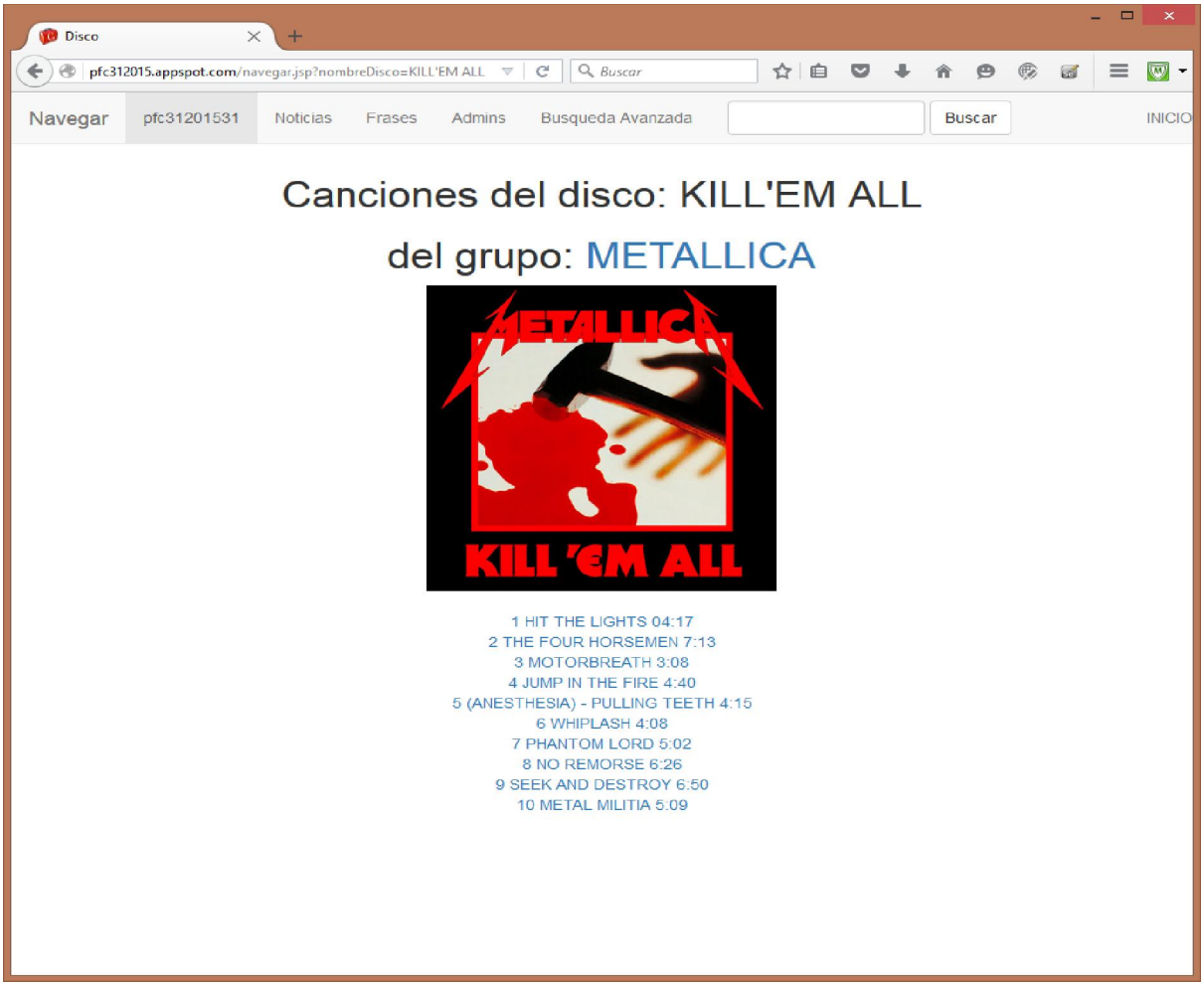
Navegar: Mostrar Grupos Carrusel…



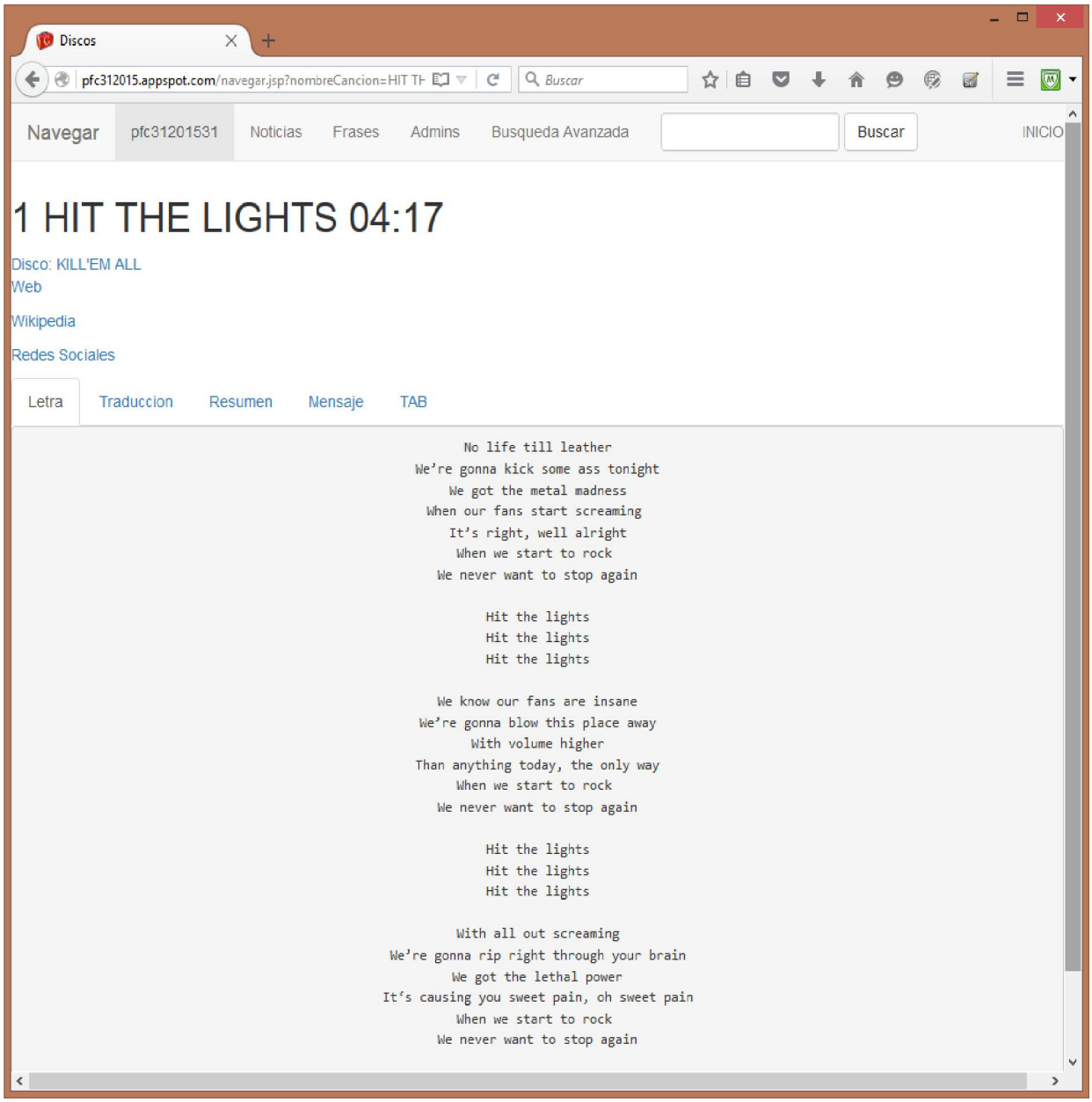
Discos de un Grupo:



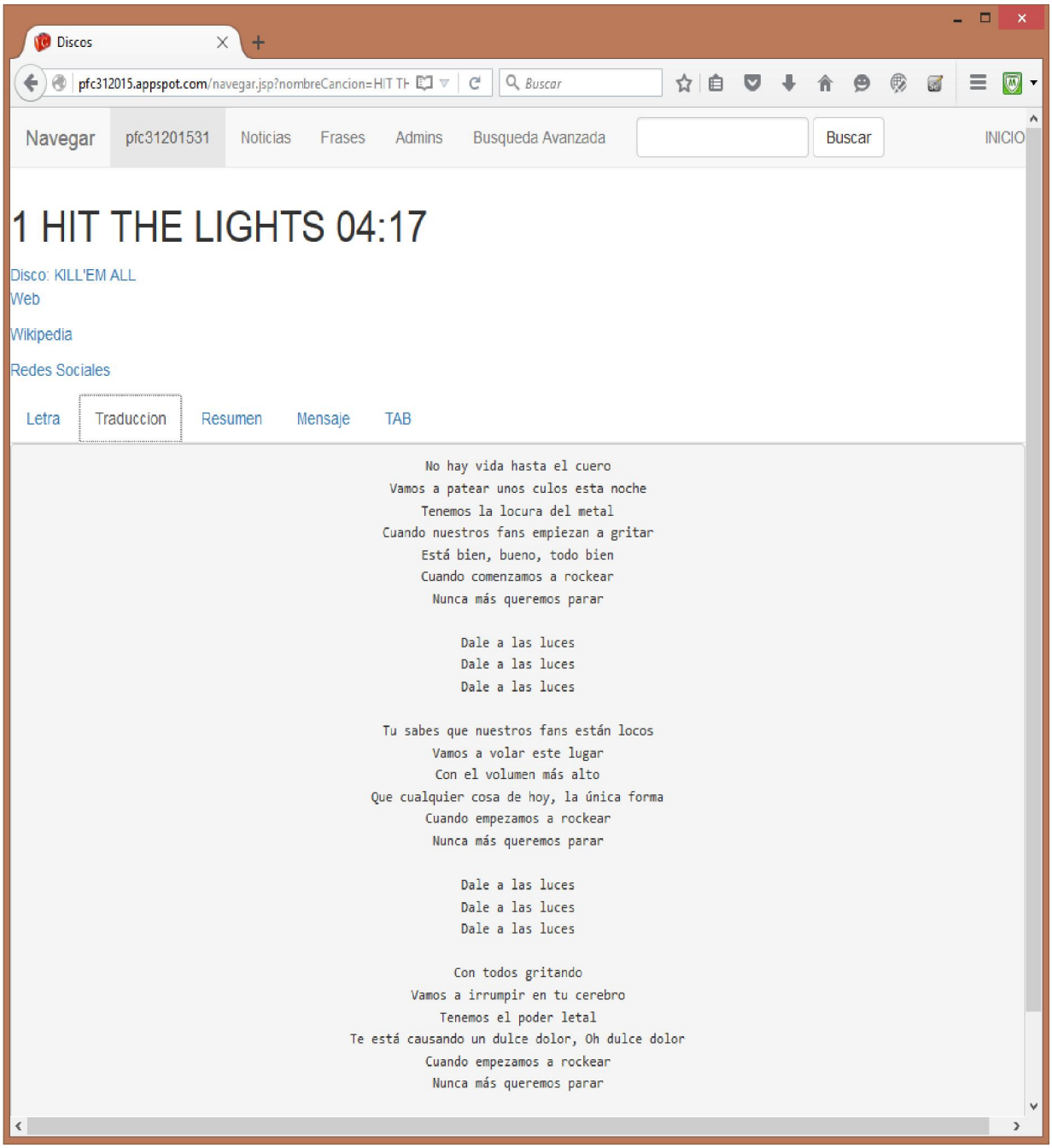
Canciones de un Disco:



Información de una Canción:

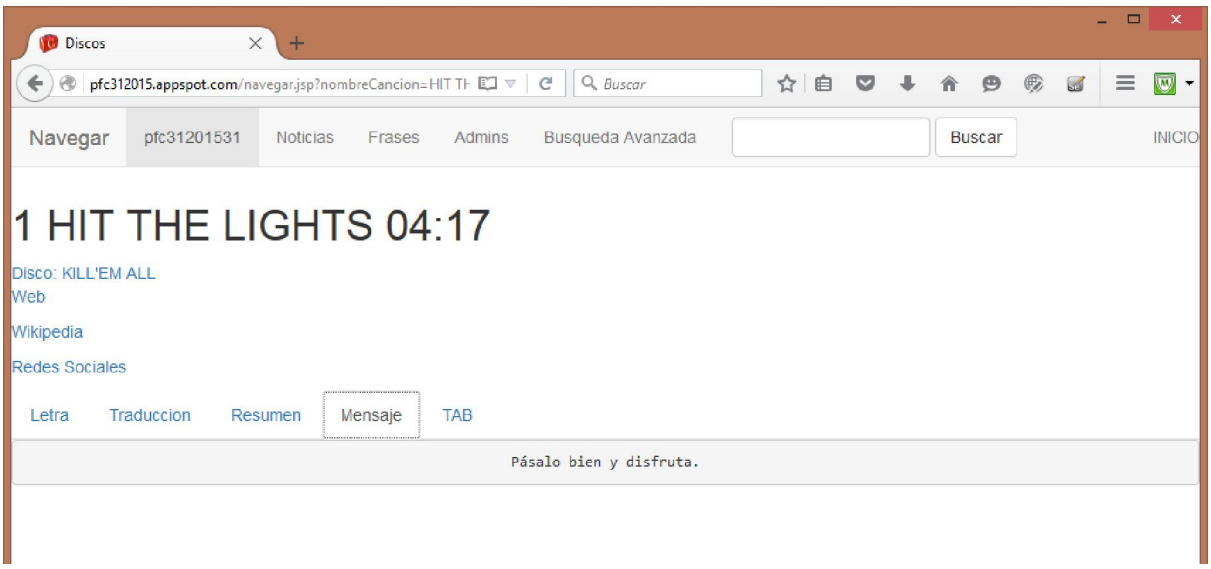


Traducción:

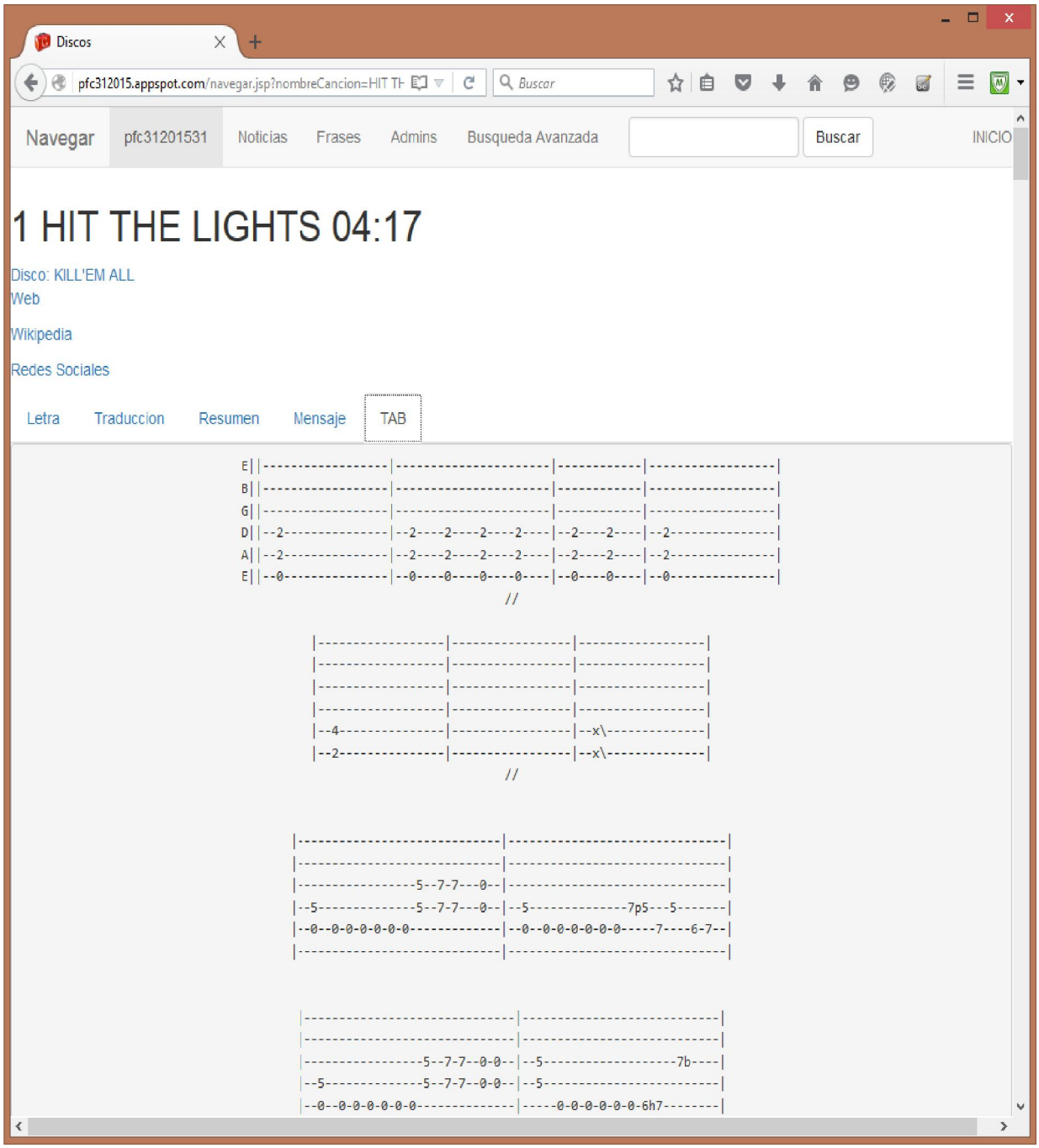

Discos
pfc312015.appspot.com/navegar.jsp?nombreCancion=HIT TH
Buscar
Navegar
pfc31201531
Noticias
Frases
Admins
Busqueda Avanzada
Buscar
INICIO
1 HIT THE LIGHTS 04:17
Disco: KILL'EM ALL
Web
Wikipedia
Redes Sociales
Letra
Traduccion
Resumen
Mensaje
TAB
No hay vida hasta el cuero
Vamos a patear unos culos esta noche
Tenemos la locura del metal
Cuando nuestros fans empiezan a gritar
Está bien, bueno, todo bien
Cuando comenzamos a rockear
Nunca más queremos parar
Dale a las luces
Dale a las luces
Dale a las luces
Tu sabes que nuestros fans están locos
Vamos a volar este lugar
Con el volumen más alto
Que cualquier cosa de hoy, la única forma
Cuando empezamos a rockear
Nunca más queremos parar
Dale a las luces
Dale a las luces
Dale a las luces
Con todos gritando
Vamos a irrumpir en tu cerebro
Tenemos el poder letal
Te está causando un dulce dolor, Oh dulce dolor
Cuando empezamos a rockear
Nunca más queremos parar

Resumen:

Mensaje:

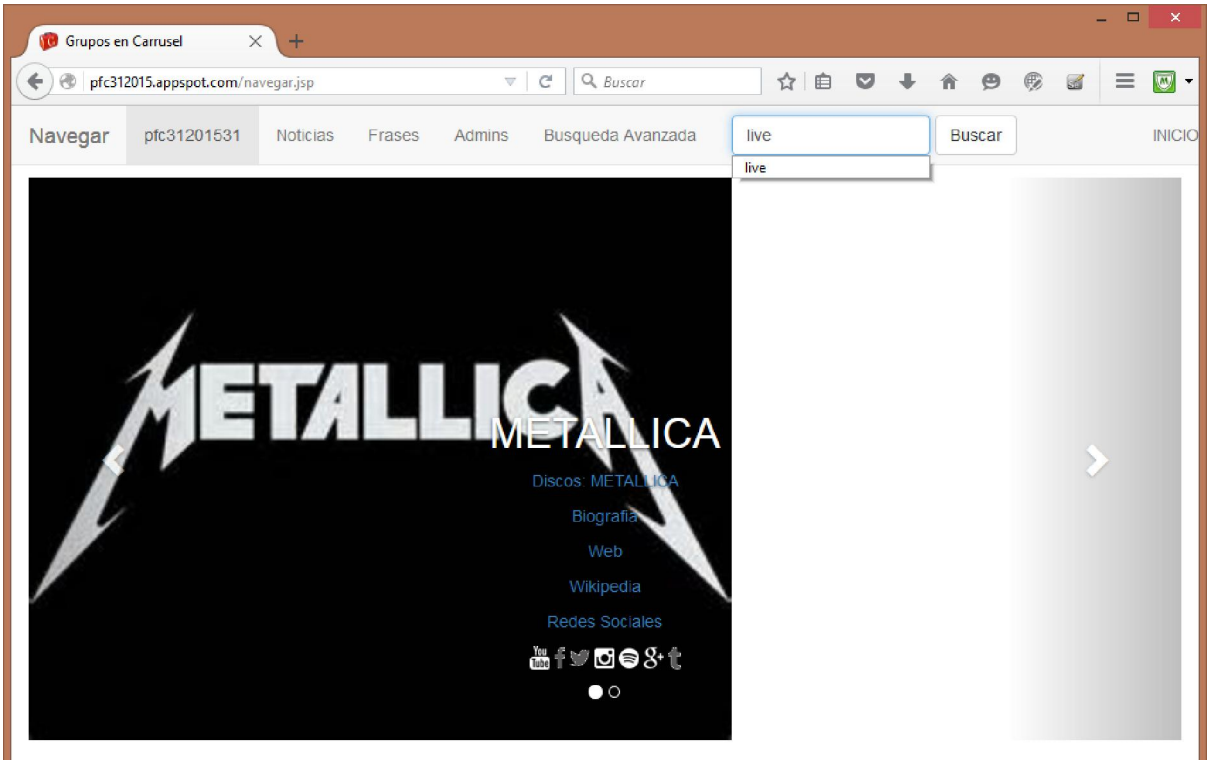


Tab para guitarra:

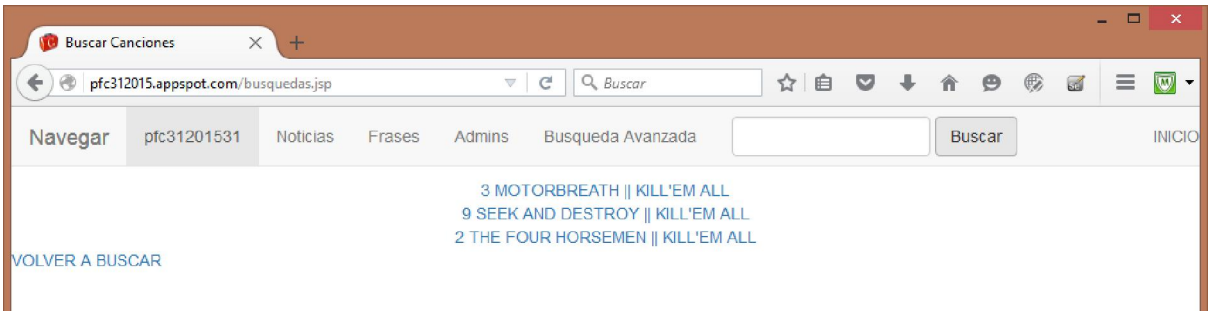


BÚSQUEDAS:

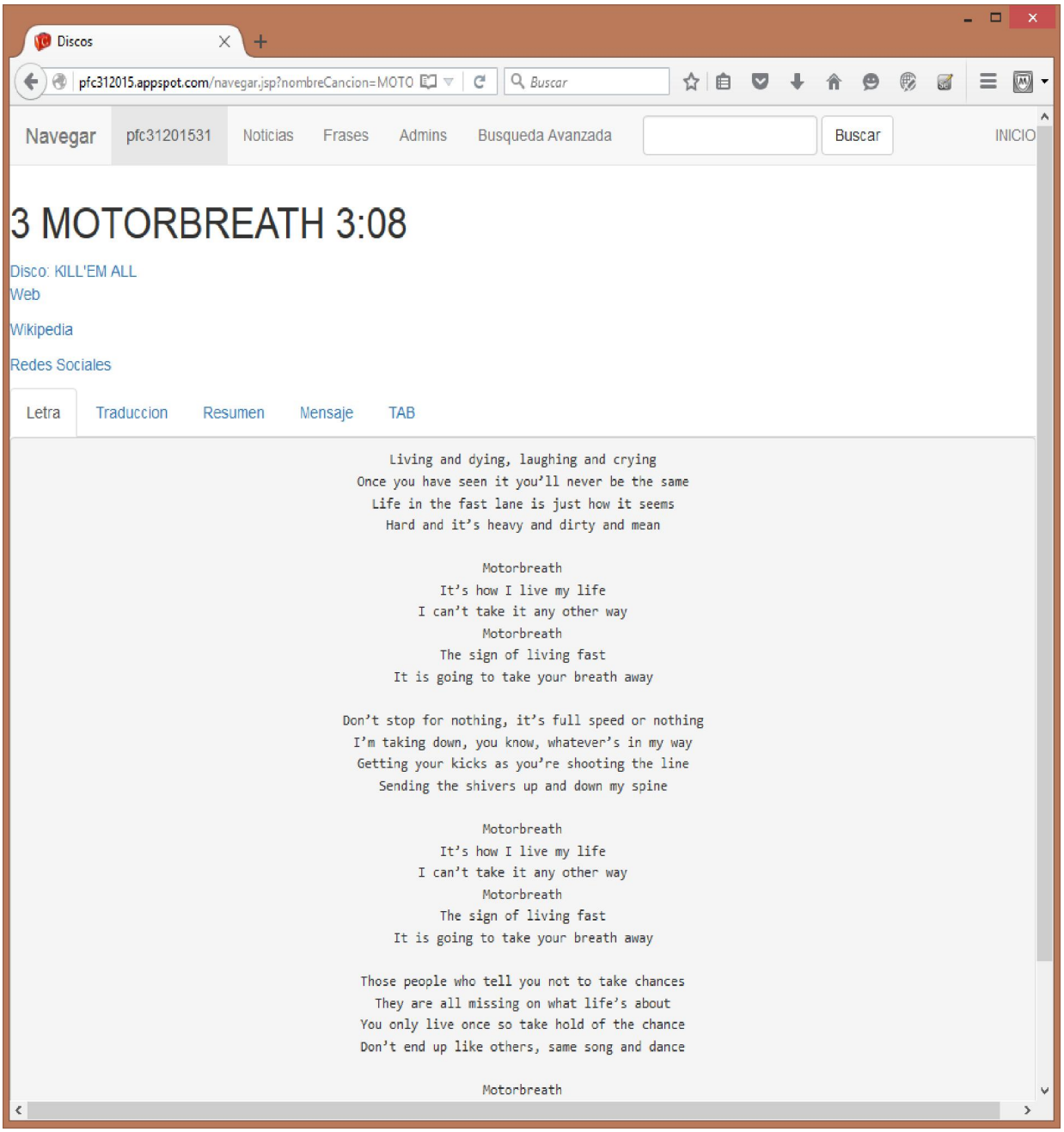
Buscar canciones que contengan en la letra el texto “live”:



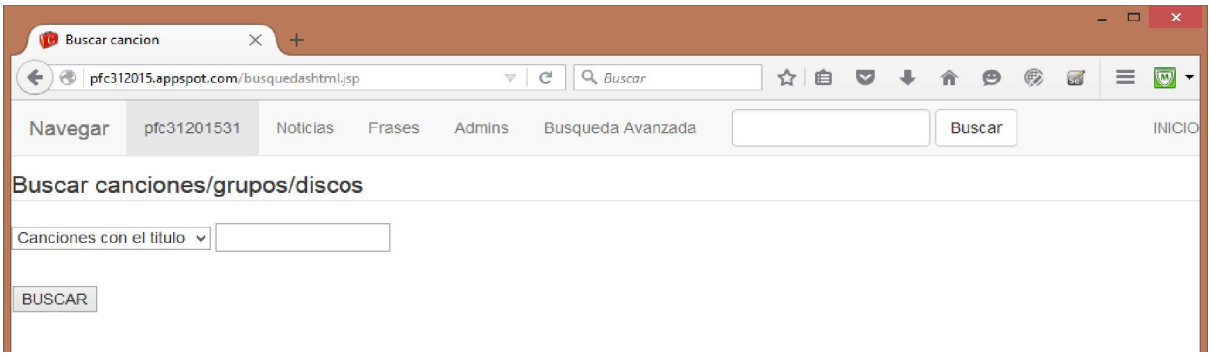
Resultado de la búsqueda de “live”:

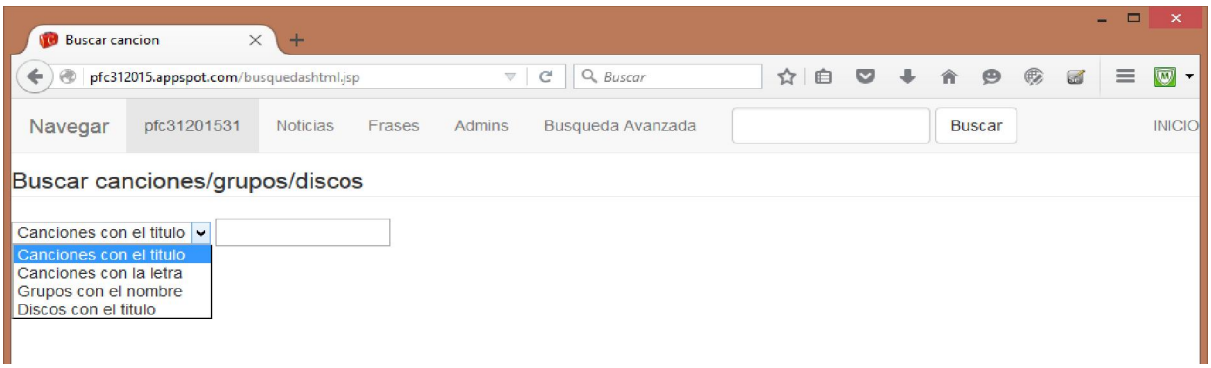


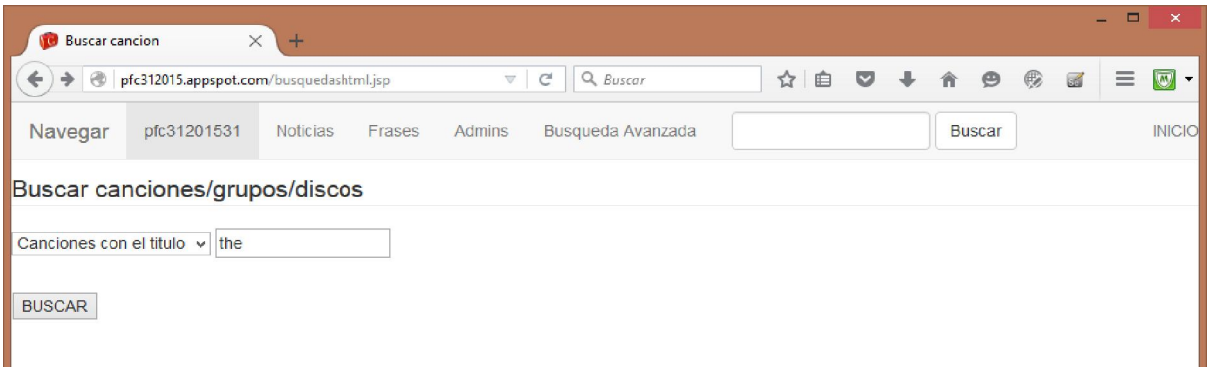
Canción a la que se accede tras la búsqueda:

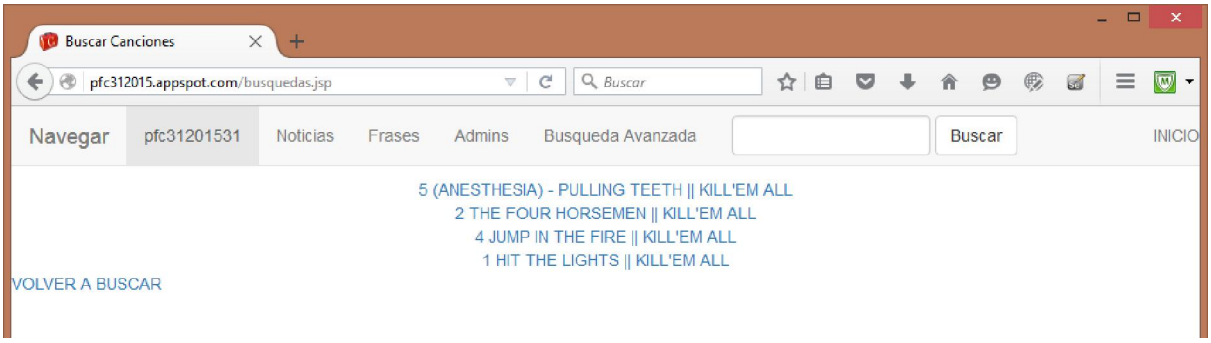


Búsqueda Avanzada:

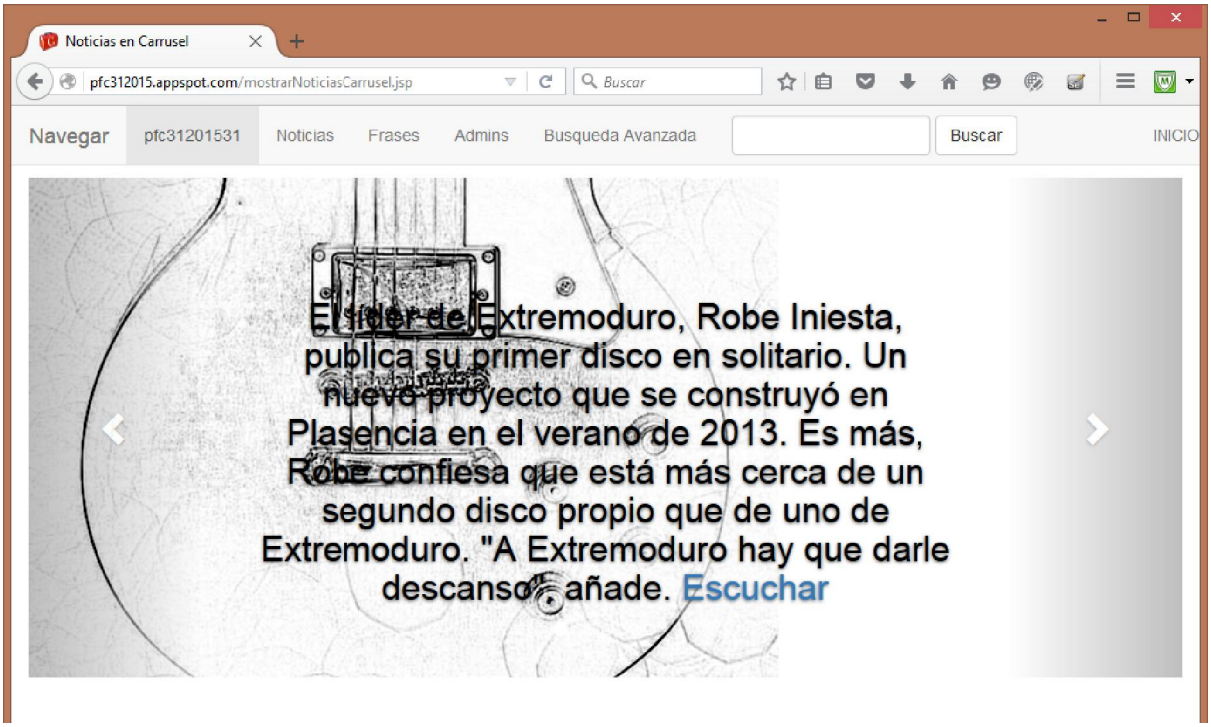




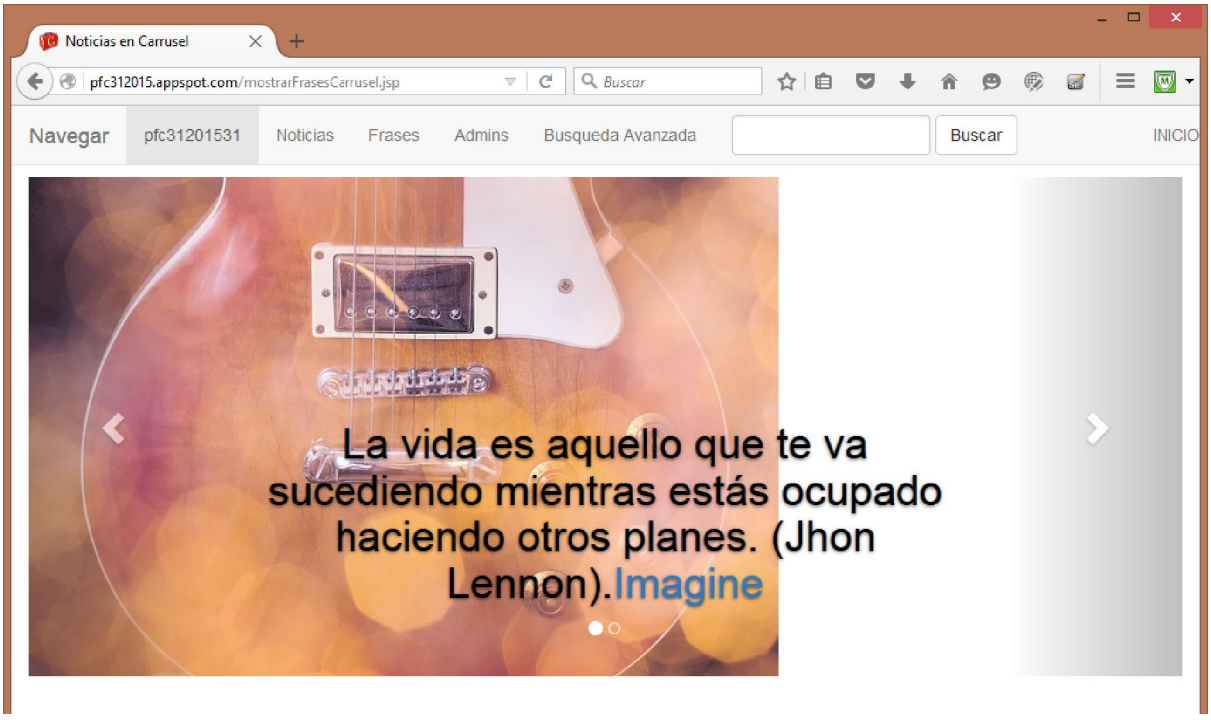




Noticias:



Frases:



ANEXO 2: VISIÓN GENERAL DE LAS CARACTERÍSTICAS DEL APP ENGINE

App Identity	A framework that provides access to the application's identity, and the ability to assert this identity using OAuth.
Blobstore	Allows your application to serve large data objects, such as video or image files, that are too large for storage in the Datastore service.
Capabilities	Detects outages and scheduled maintenance for specific services, so that an application may bypass them or notify users.
Channel	Creates a persistent connection between your application and JavaScript clients, so you can send messages in real time without polling.
Datastore	A schemaless object datastore, with scalable storage, a rich data modeling API, and an SQL-like query language.
Datastore Backup/Restore	Allows you to export or import data to or from your application's Datastore using the Admin Console.
Dedicated Memcache	Provides a fixed cache capacity assigned exclusively to your application.
Go Runtime	Build your application in the Go programming language.
Google Cloud Endpoints	Generates APIs for Android, iOS, and web clients, making it easier to create a web backend for your app.
Google Cloud SQL	A fully-managed web service that allows you to create, configure, and use relational databases that live in Google's cloud.
Google Cloud Storage Client Library	Read and write to Google Cloud Storage, with internal error handling and retry logic.
HRD Migration Tool	Migrates application data stored in the Blobstore or the deprecated Master/Slave Datastore into the GA High Replication Datastore.
Images	Manipulate, combine, and enhance images. Converts between image formats, access image metadata such as height and frequency of colors.
Java Runtime	Build your application in the Java programming language.
Logs	Programmatic access to application and request logs from within your application.
Mail	Send email messages on behalf of administrators and users with Google Accounts, and receive mail at various addresses.
MapReduce	An optimized adaptation of the MapReduce computing model for efficient distributed computing over large data sets.

Memcache	A distributed, in-memory data cache that can be used to greatly improve application performance.
Modules	Factor applications into logical components that can share stateful services and communicate in a secure fashion.
Multitenancy	Makes it easy to compartmentalize your data to serve many client organizations from a single instance of your application.
OAuth	Uses the OAuth protocol to provide a way for your app to authenticate a user who is requesting access without asking for credentials (username and password)
PHP Runtime	Build your application in the PHP programming language.
Python Runtime	Build your application in the Python programming language.
Remote	Access App Engine services from any application. For example, access a production datastore from an app running on your local machine.
Scheduled Tasks	Configure tasks that run at defined times or regular intervals.
Search	Perform Google-like searches over structured data such as: plain text, HTML, atom, numbers, dates, and geographic locations.
SendGrid	Use SendGrid's library to send emails from your app and you can see statistics on opens, clicks, unsubscribes, spam reports and more.
Sockets	Supports outbound sockets using the language-specific, built-in libraries.
SSL for Custom Domains	Serve applications via HTTPS and HTTP from a custom domain rather than an appspot.com address.
Task Queue	Allows applications to perform work outside of a user request, using small, discrete tasks, that are executed later.
Task Queue REST API	Enables the use of an App Engine task queue over REST.
Task Queue Tagging	Leases up to a specified number of tasks with the same tag from the queue for a specified period of time.
Traffic Splitting	Routes incoming requests to different versions of your app, allowing you to do A/B testing and roll out new features incrementally.
Twilio	Enables your application to make and receive phone calls, send and receive text messages, and make VoIP calls from any phone, tablet, or browser.
URL Fetch	Uses Google's networking infrastructure to efficiently issue HTTP and HTTPS requests to URLs on the web.
Users	Allows applications to sign in users with Google

	Accounts or OpenID, and address these users with unique identifiers.
XMPP	Allows an application to send and receive chat messages to and from any XMPP-compatible chat messaging service.

Ilustración 5-2 Características del GAE

ANEXO 3: TABLATURA, TAB, TABLATURE[liii]

Una tablatura, o sólo TAB, es una forma de notación musical que en lugar de indicar las notas, indica la posición en el instrumento para la interpretación de una pieza para instrumentos de cuerda, como la tan popular guitarra y el menos preciado bajo.

Utilizar tablaturas es una sencilla y rápida manera de aprender canciones, y ahí radica la cierta polémica de algunos músicos por creer que son "un atajo", que no fomentan el aprendizaje, aun cuando tienen siglos de creación. Obviamente jamás remplazaran lo que provee una partitura (como los tiempos de la notas), pero para quien toca la guitarra como un pasatiempo, lo considero un recurso válido.

La tablatura

```
E|--------------------------------| Primera cuerda
B|--------------------------------| Segunda cuerda
G|--------------------------------| Tercera cuerda
D|--------------------------------| Cuarta cuerda
A|--------------------------------| Quinta cuerda
E|--------------------------------| Sexta cuerda
```

Las líneas indican las cuerdas de la guitarra, de la más aguda a la más grave. Ver vídeo de ayuda[liv].

La letras de la izquierda indican el tono en que deben estar afinadas, aunque esto se suele mencionar en el comienzo de los archivos, después del nombre y canción del artista. Como la mayoría de las tablaturas que encontramos en Internet están en sitios anglosajones, se utiliza el sistema de notación inglesa, en el caso de arriba la afinación está en Mi (E stándar), esto es:

A - La
B - Si
C - Do
D - Re
E - Mi
F - Fa
G - Sol

Veamos un ejemplo más interesante:

```
|-----------------|-----------------|
|-----------------|-----------------|
|-----0-----------|-----0-----------|
|---4---4---4---4-|---4---4---4---4-|
|-2-------2---5---|-0-------0---5---|
|-----------------|-----------------|
```

El número indica el traste en que se deberá posicionar el dedo de la mano izquierda (si eres diestro), en el ejemplo sería: traste 2 de la quinta cuerda, traste 4 de la cuarta cuerda y tercera cuerda al aire (el cero indica que no se debe presionar ningún traste).

Acordes

```
|-2--2-2-3---2-2-0---2-|-0------0-------|
|-3--3-3-3---3-3-3---3-|-1------2-------|
|-2--2-2-2---2-2-2---2-|-0------2-------|
|-0--0-0-0---0-0-0---0-|-2------2-------|
|----------------------|-3------0-------|
|----------------------|----------------|
 D Dsus4 D Dadd2 D C A
```

Cuando encontramos un número encima de otro generalmente describirá un acorde, en ocasiones puede ser parte de una segunda guitarra. Para interpretarlo colocamos los dedos según nos indique y tocamos las cuerda a la vez. Puede ocurrir la situación en que se nos indica el nombre del acorde, el primero en el ejemplo es Re mayor (D).

Símbolos y técnicas

Existen varias técnicas en la guitarra, en las tablaturas hay ciertos "símbolos" para representarlas, suelen variar de un archivo a otro pero el mismo autor explica su significado. Los básicos y más frecuentes:

```
|-------5b7b5-------| Bend (up/down)
|-------5h7---------| Hammer-on
|-------7p5---------| Pull-off
|-------7~----------| Vibrato
|-------7s5---------| Slide
```

Efectos

- **Hammer-on:** Se hace tocando la cuerda e instantáneamente se golpea un traste posterior al que se está pisando o si se está tocando la cuerda al aire.
- **Pull-off:** En éste se hace casi lo mismo que con el Hammer-on, sólo que en vez de golpear un traste posterior se deja un dedo en un traste anterior y, al tocar la cuerda, se suelta el traste sin quitar el dedo del traste anterior, prácticamente pellizcando la cuerda con el dedo que se quita.
- **Bend:** Al tocar una cuerda se hace presión sobre el traste con el dedo, así que el bend consiste en mantener la presión sobre ese traste y elevar la cuerda sin soltar el traste. (No puede hacerse tocando una cuerda al aire).
- **Slide:** Al tocar la cuerda se arrastra el dedo que se encuentra sobre el respectivo traste hasta otro, ya sea posterior o anterior al que se está pisando, sin quitar el dedo. (No puede hacerse tocando la cuerda al aire).
- **Nota muerta:** Se hace tocando la cuerda en el lugar del respectivo traste sin hacer presión sobre él.
- **Vibrato:** Se toca la nota y se hace vibrar con el vibrato o palanca de la guitarra, o bien haciendo un bend lo más rápido posible hacia arriba y hacia abajo.

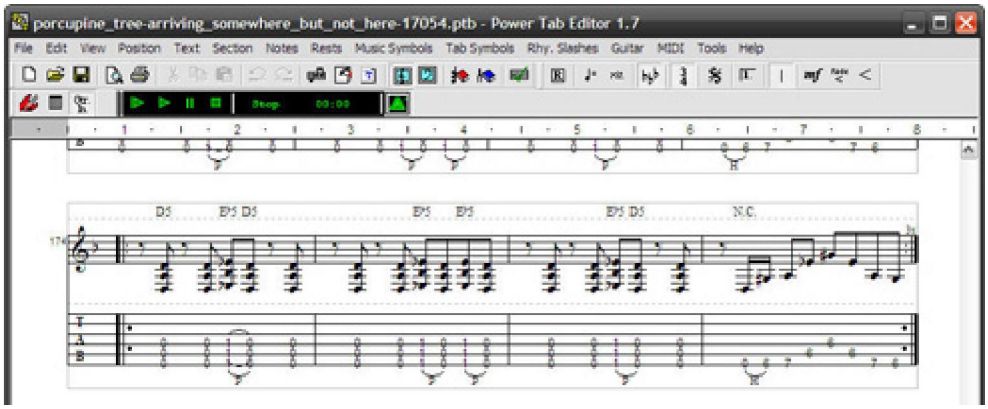
- **Palm mute:** La nota es ligeramente apagada con el filo de la mano que hace el pick mientras esta está tocando las notas.

Programas para crear tablaturas:

Existen distintas aplicaciones para crear y visualizar tablaturas. El formato más popular es un simple .txt, sin embargo existen otros más atractivos.

Un programa para visualizar, editar y escribir tablaturas en formato de texto (.txt), tiene que tener un buscador de sitios con este tipo de archivos.

PowerTab (.ptb). Es uno de los más populares y mi preferido, ¿será por qué es gratuito? La ventaja que tiene este tipo de programa es poder ver al mismo tiempo la partitura y tablatura. Además, una de sus mejores características es reproducir en MIDI la notación, así no tienes que estar escuchando la canción original. Windows.



Guitar Pro (.gp3, .gp4, .gp5). Una aplicación de pago muy semejante a PowerTab, puedes ver la tablatura y la partitura al mismo tiempo, tal vez en esta sea más rápido y sencillo editar una tablatura. Mac, Windows.

TuxGuitar (.ptb, .gp3, .gp4, .gp5). La ventaja de este programa es poder leer archivos .ptb y gp*. No le he usado así que no puedo describirlo del todo, pero parece una buena opción como "visor" tipo Adobe Reader, pero para tablaturas. Linux.

El día que tocar las canciones de tus grupos favoritos fue ilegal:

En el 2006 la MPA empezó a tomar acciones legales contra los sitios que ofrecían tablaturas de forma gratuita (incluso los que no tenían publicidad), ellos reclamaban que los artistas no recibían ninguna regalía en compensación. Algunos sitios se trataron escudar bajo el derecho del fair use, afirmando que sus fines eran totalmente educativos, no fue suficiente, mucho de ellos cerraron.

Dónde conseguir tablaturas:

Por fortuna existen varios sitios, independientemente de lo que ocurrió en el 2006, con una larga base de datos en la que podemos buscar tablaturas.

Ultimate Guitar. Una de la mayor base de datos, las tablaturas están bien organizadas como txt, ptb o gp*.

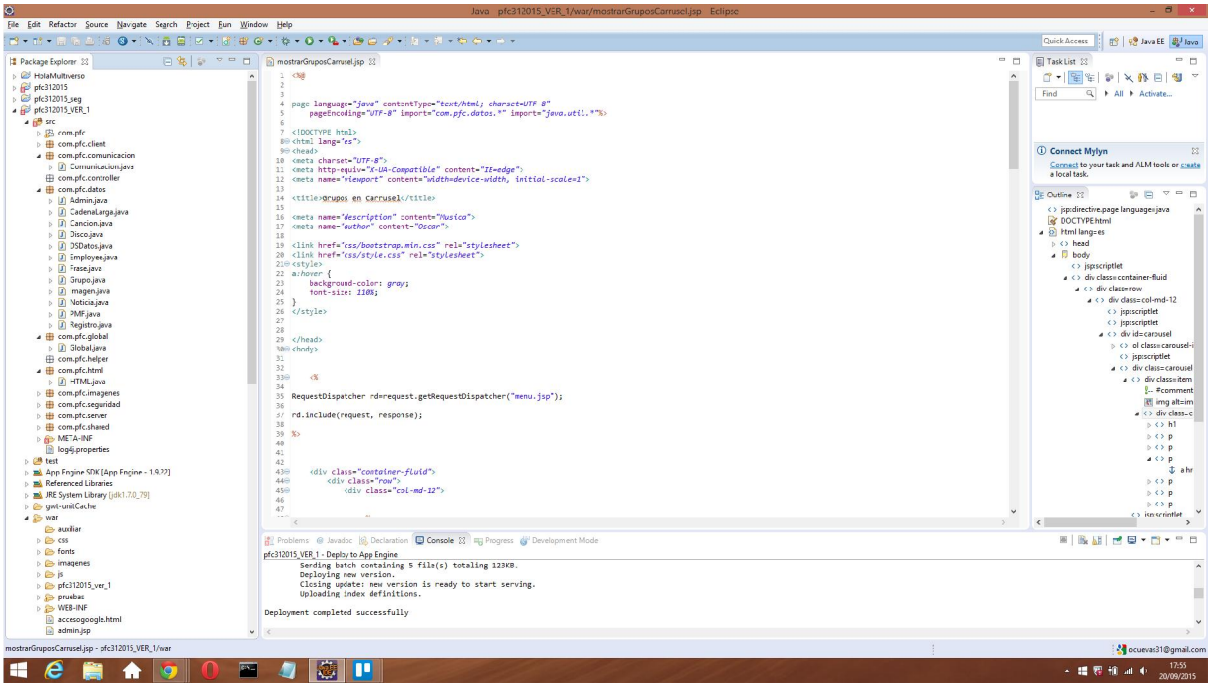
La cuerda.net. Aquí podemos encontrar canciones en español. Sin embargo, frecuentemente sólo son las letras de las canciones con los acordes que la acompañan.

MXTabs. El “primer” sitio en el cual se llegó a un acuerdo con los artistas para poder publicar las tablaturas.

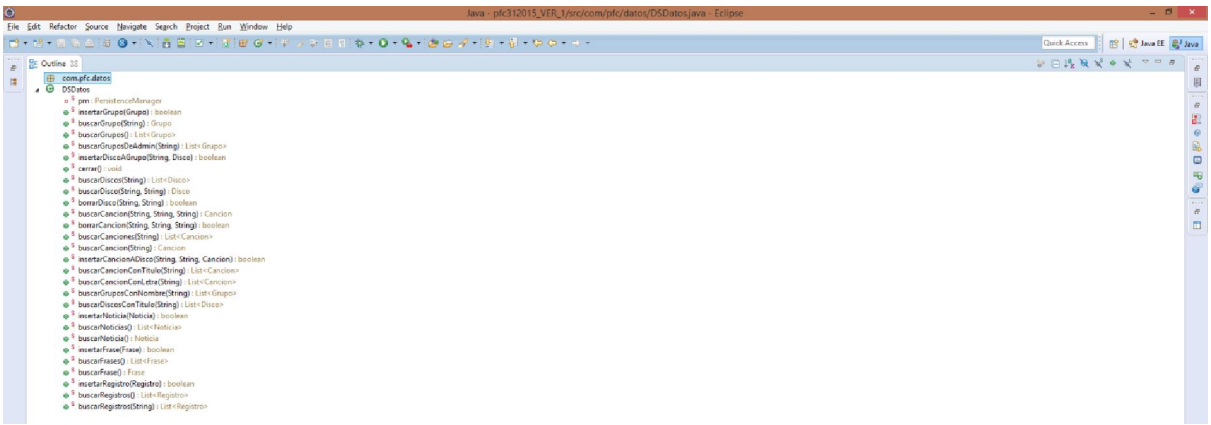
GProTab. Una página que provee tablaturas para guitar pro, se encuentra en Rusia.

ANEXO 4: UN POCO DE "CÓDIGO"

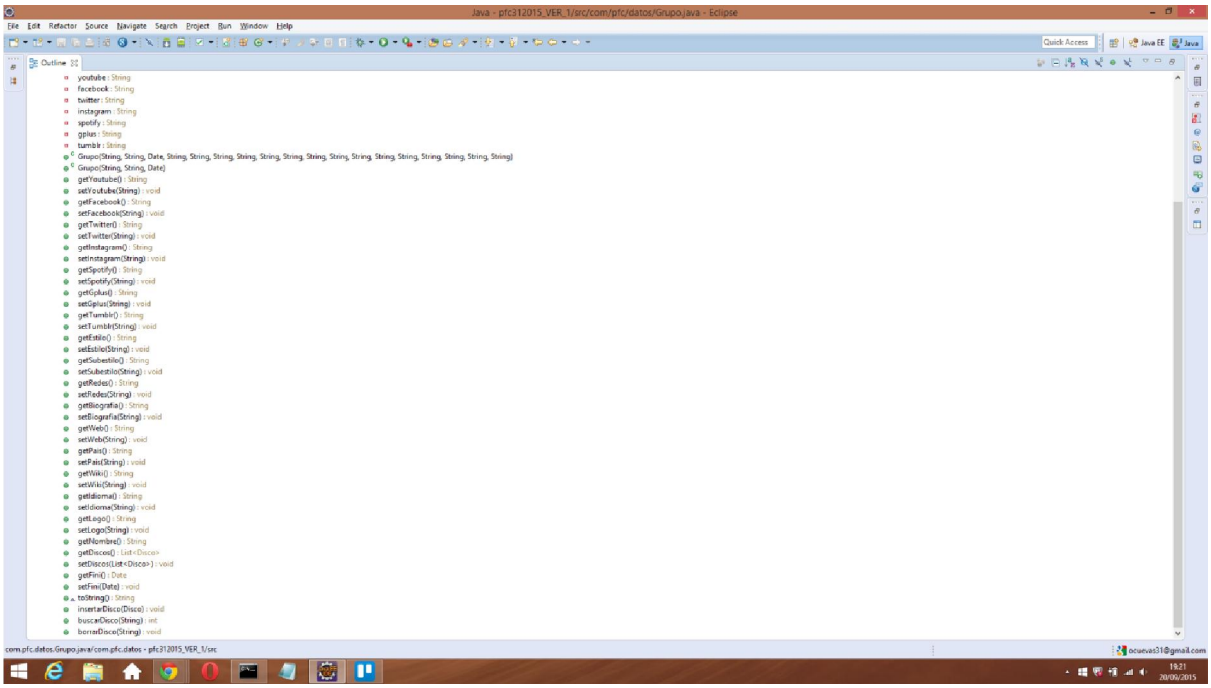
En la siguiente ilustración se muestran las clases Java y la estructura principal del código del PFC.



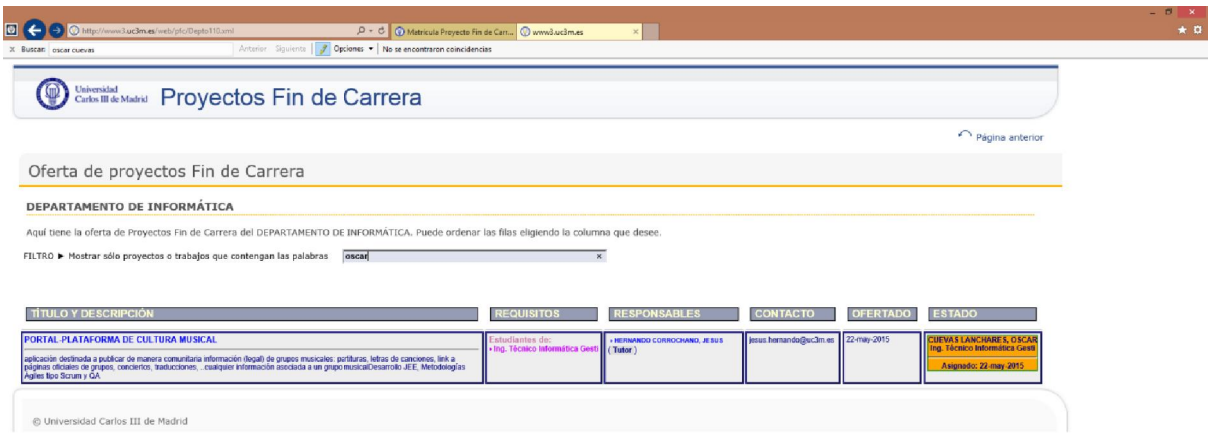
La clase DSDatos es la encargada del acceso al Data Store de GAE. En la ilustración se muestran los métodos (operaciones) que son necesarias para el tratamiento de los datos de la aplicación.



Ahora se muestra (a modo de ejemplo) la clase Grupo, con sus métodos.



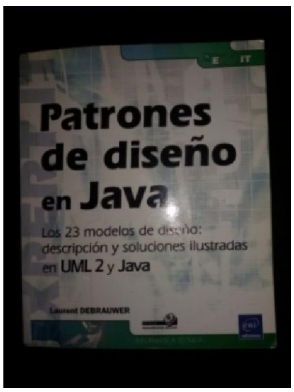
El presente PFC en el tablón de PFC de una Universidad Carlos III (UC3M)

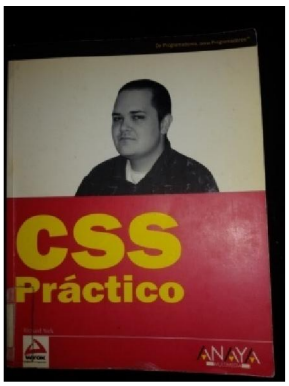


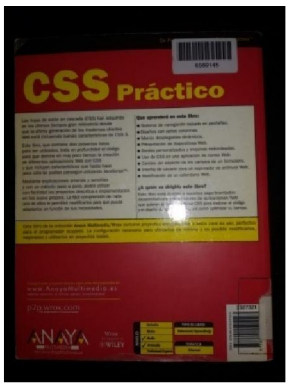
BIBLIOGRAFÍA

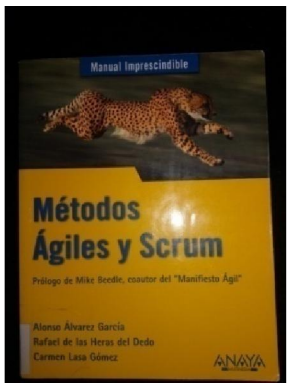
LIBROS EN PAPEL:

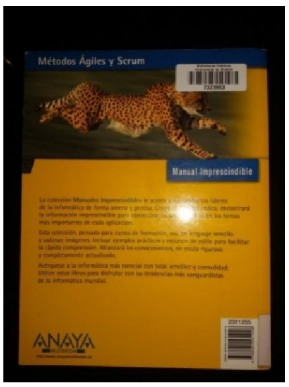
- DEB pat. Patrones de diseño en Java. Los 23 modelos de diseño: descipcion y soluciones ilustradas en UML2 y Java. Laurent Debrauwer. Ediciones Eni.
- ALV man. Métodos Ágiles y Scrum. Alonso Álvarez GarcÍa, Rafel de las Heras del Dedo, Carmen Lasa Gómez. Anaya multimedia.
- HTM luc.Apps HTML5 para móviles. Desarrollo de aplicaciones para smartphones y tablets basado en tecnologías web. Damián de Luca. Marcombo.
- YOR css. CSS práctico. Richard York. Anaya multimedia.
- HTM dav. HTML5 Matthew David. Anaya multimedia.

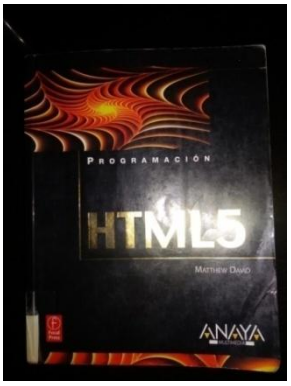


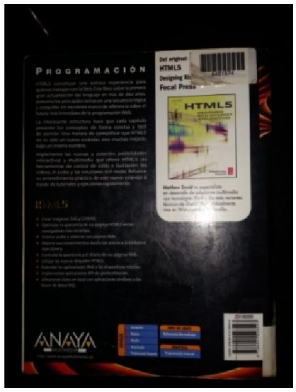




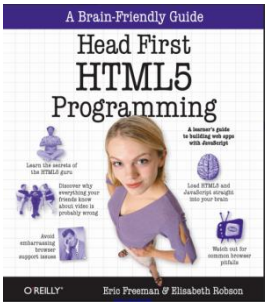


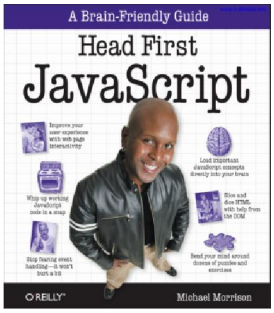


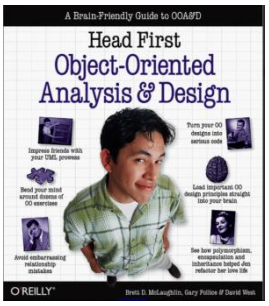




LIBROS EN PDF:







INTERNET:

RAE. Real Academia Española.
http://www.rae.es/

Google. (2015). Google Developers Console. Disponible en:
https://console.developers.google.com

Google Developers. (2014). Blobstore Java API Overview. Disponible en:
https://developers.google.com/appengine/docs/java/blobstore

Universidad Carlos III de Madrid. (2010). Plantilla para presupuestos de Proyectos de Fin de Carrera y Trabajos de Fin de Grado. Fecha de consulta: 15 de Septiembre de 2014. Disponible en:
http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx

DATA STORE:

https://cloud.google.com/appengine/docs/java/storage
https://cloud.google.com/appengine/docs/java/datastore/
http://www.adictosaltrabajo.com/tutoriales/datastore-jdo/
https://www.youtube.com/watch?v=_P1wcF_XBIE
https://www.youtube.com/watch?v=fQazhzcC-rg
http://rominirani.com/2009/12/18/episode-13-using-the-blobstore-java-api/
http://blog.mbonell.com/2011/09/manejo-de-imagenes-en-google-app-engine-java-blobstore/
http://blog.mbonell.com/2011/09/manejo-de-imagenes-en-google-app-engine-java-blobstore/
https://cloud.google.com/appengine/docs/java/blobstore/

EJEMPLO COMPLETO de servlet, jsp y GAE:

https://ingeniods.wordpress.com/2011/11/23/mi-primera-aplicacion-en-google-app-engine/
http://www.lab.inf.uc3m.es/~a0080802/RAI/servlet.html

OAUTH:

https://developers.google.com/identity/sign-in/web/sign-in#before_you_begin
https://developers.google.com/identity/sign-in/web/
https://developers.google.com/identity/sign-in/web/
https://developers.google.com/identity/sign-in/web/backend-auth

CORREO APP ENGINE:

https://cloud.google.com/appengine/docs/java/mail/usingjavamail
http://stackoverflow.com/questions/13854037/send-mail-to-multiple-recipients-in-java

CACHÉ:

https://www.jcea.es/artic/web-cache.htm
http://www.hellogoogle.com/tutorial-cache-web/
http://stackoverflow.com/questions/3055268/add-an-expires-or-a-cache-control-header-in-jsp

GWT (aunque al final no se ha utilizado):

http://www.paradigmatecnologico.com/blog/entendiendo-que-es-gwt/
https://es.wikipedia.org/wiki/Google_Web_Toolkit
http://samples.gwtproject.org/samples/Showcase/Showcase.html#!CwCheckBox
http://www.gwtproject.org/doc/latest/DevGuideUiCss.html

ANÁLISIS APP ENGINE (GAE):

http://unadocenade.com/una-docena-de-ventajas-de-usar-google-app-engine-como-infraestructura-tecnologica/
http://stackoverflow.com/questions/3444911/alternative-for-google-appengine
https://www.heroku.com/
http://www.one.com/es/
http://www.appscale.com/
https://code.google.com/p/typhoonae/
https://cloud.google.com/appengine/docs
http://www.digiworks.es/blog/2012/07/31/que-es-google-app-engine-y-para-que-sirve/
https://es.wikipedia.org/wiki/Google_App_Engine
https://cloud.google.com/appengine/docs/whatisgoogleappengine
https://cloud.google.com/appengine/features/
https://cloud.google.com/appengine/docs/java/

VARIOS:

https://myspace.com/
http://play.spotify.com
https://itunes.apple.com

CUESTIONES LEGALES:

http://www.baquia.com/tecnologia-y-negocios/entry/emprendedores/letras-de-canciones-otra-pelea-de-los-derechos-de-autor
http://www.nytimes.com/2010/05/10/business/media/10lyrics.html?ref=technology&_r=0
http://www.gracenote.com/
http://www.lyricfind.com/
http://www.literautas.com/es/blog/post-5712/los-derechos-de-autor-y-los-escritores/
http://biblioguias.unex.es/content.php?pid=504133&sid=4147465
http://www.metallica.com/help/copyright-faq.asp

SCRUM Y ÁGILES:

http://www.javiergarzas.com/2014/09/scrum-master-consideraciones-importantes.html

NOTAS

[i] https://es.wikipedia.org/wiki/Tablatura
[ii] https://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_n%C3%BAmero_de_usuarios_de_Internet
[iii] http://noticiascausayefecto.com/2015-comenzo-con-mas-de-3-mil-millones-de-usuarios-de-internet-2/
[iv] https://jcp.org/en/jsr/detail?id=369
[v] http://download.oracle.com/otndocs/jcp/servlet-3_1-fr-eval-spec/index.html
[vi]https://cloud.google.com/
[vii] https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRX_3-HFa_YbGPIxMcaCRWXUMEgq3Lerz1zt-wJS29lVSmHk0rm4g
[viii] http://www.adictosaltrabajo.com/tutoriales/cloudcomputing/
[ix] http://www.adictosaltrabajo.com/tutoriales/cloudcomputing/
[x] http://www.ibm.com/cloud-computing/es/es/landing/bluemix.html?&csr=4Q15+IOT+Cloud+Paid+Search+ES+DEV&iio=pcloud&S_PKG=&S_TACT=C4280B0W&jm=&cmp=C4280&ct=C4280B0W&cr=google&cm=k&ccy=us&ck=bluemix&cs=p&cn=IBM+Bluemix&mkwid=s5VtrKms8-dc_47757532770_432dhv5077
[xi] https://es.wikipedia.org/wiki/Docker_(software)
[xii] http://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades
[xiii] https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/
[xiv] https://es.wikipedia.org/wiki/Node.js
[xv] http://www.maestrosdelweb.com/que-es-javascript/
[xvi] http://www.desarrolloweb.com/articulos/1557.php
[xvii] https://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-es-rest/
[xviii] https://raiolanetworks.es/blog/que-es-bootstrap/
[xix] https://es.wikipedia.org/wiki/JavaServer_Faces
[xx] https://es.wikipedia.org/wiki/Java_Persistence_API
[xxi] http://globalmentoring.com.mx/curso-javaee/
[xxii] http://www.javiergarzas.com/2014/09/scrum-master-consideraciones-importantes.html
[xxiii] http://spanishpmo.com/index.php/los-12-principios-del-manifiesto-agil/
[xxiv] http://www.marblestation.com/?p=661
[xxv] http://www.javiergarzas.com/wp-content/uploads/2014/03/rol_scrumMaster.jpg
[xxvi] http://www.desarrolloweb.com/articulos/roles-scrum.html
[xxvii] http://www.javiergarzas.com/2014/02/rol-del-product-owner.html
[xxviii] https://developers.google.com/identity/sign-in/web/sign-in#before_you_begin
[xxix] http://www.thegameofcode.com/2012/07/conceptos-basicos-de-oauth2.html
[xxx] http://www.oracle.com/technetwork/java/index.html
[xxxi] https://www.oracle.com/sun/index.html?PHPSESSID=8fd76773d26875481e097a4a27c7a6a1
[xxxii] http://www.oracle.com/es/index.html
[xxxiii] https://eclipse.org/
[xxxiv] https://developers.google.com/eclipse/
[xxxv] https://youtu.be/GGJC_i7Dw6c
[xxxvi] https://cloud.google.com/appengine/docs/java/datastore/jdo/overview-dn2
[xxxvii] http://www.latinuxpress.com/books/drafts/web2py/caps/cap1.html
[xxxviii]https://cloud.google.com/appengine/docs/quotas?hl=es&_ga=1.19635697.878196939.1439205492#Safety_Quotas_and_Billable_Quotas
[xxxix] https://console.developers.google.com/datastore?project=pfc312015&queryType=kind&ns=&kind=__ALL_KINDS__
[xl] https://console.developers.google.com/datastore/stats?project=pfc312015&namespace=__all__&kind=Grupo
[xli] https://console.developers.google.com/datastore/query?project=pfc312015&namespace=&kind=Grupo&queryType=KindQuery
[xlii] http://www.kabooza.com/globalsurvey.html
[xliii] https://console.developers.google.com/traces/tasks/4405545224241675126?project=pfc312015
[xliv] https://appengine.google.com/
[xlv] http://ronjeffries.com/
[xlvi] http://trello.com
[xlvii] http://www.seleniumhq.org/
[xlviii] https://appengine.google.com/billing/history?&app_id=e~pfc312015&version_id=1.389023865326674088
[xlix] https://appengine.google.com/adminlogs?&app_id=e~pfc312015&version_id=1.388860613760430060
[l] https://es.wikipedia.org/wiki/Bot
[li] https://www.google.es/adsense/start/#?modal_active=none
[lii] http://www.fatdux.com/es/what/what-is-ux
[liii] http://www.blogoff.es/2008/10/06/aprendiendo-a-tocar-canciones-con-tablaturas-como-crearlas-y-donde-conseguirlas/
[liv] https://www.youtube.com/watch?v=w1PogdK1JVc